НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр КОВАЛЬ
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»
на тему: «Інструментальні засоби формування бази даних показників інноваційної діяльності провінції Шаньдун»

Виконав:
студент IV курсу, групи ТМ-42
Чередниченко Борис Ігорович _____
Керівник:
доцент, кандидат економічних наук
Гусєва Ірина Ігорівна _____
Рецензент:
доцент, кандидат технічних наук
Веремійчук Юрій Андрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Чередниченко Борис Ігорович_____
(прізвище, ім’я, по батькові)

1. Тема роботи Інструментальні засоби формування бази даних показників інноваційної діяльності провінції Шаньдун_____

_____

керівник роботи_____к.е.н., доцент Гусєва Ірина Ігорівна_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 202__р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи мова програмування C#, мова програмування TSQL, середовище розробки Microsoft Visual Studio, система керування базами даних MSSQL_____
_____4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_____Аналіз задачі розробки програмного

забезпечення формування бази даних показників інноваційної діяльності провінції Шаньдун. Аналіз та оцінка існуючих рішень. Вибір програмних засобів реалізації програмного рішення. Розробка програмного комплексу формування бази даних показників інноваційної діяльності_провінції_Шаньдун

5. Перелік ілюстративного матеріалу

1. Актуальність теми 2. Проблематика дипломної роботи 3. Мета та задачі 4. Постановка задачі 5. Опис програмної реалізації системи 6. Методика роботи користувача з програмою 7. Висновки

7. Дата видачі завдання "11" жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	02.02.20 – 12.02.20	
2.	Вивчення та аналіз задачі	19.02.20 – 26.02.20	
3.	Розробка архітектури та загальної структури системи	03.03.20 – 11.03.20	
4.	Розробка структур окремих підсистем	21.03.20 – 04.04.20	
5.	Програмна реалізація системи	08.04.20 – 25.04.20	
6.	Оформлення пояснювальної записки	27.04.20 – 31.04.20	
7.	Захист програмного продукту	03.05.20 – 20.05.20	
8.	Передзахист	08.06.20 – 13.06.20	
9.	Захист	15.06.20 – 19.06.20	

Студент _____ Чередниченко Б.І.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Гусєва І.І.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

В роботі розглянуті теоретичні та практичні аспекти розробки програмного продукту для формування бази даних показників інноваційної діяльності провінції Шаньдун.

Метою роботи є розробка автоматизованої системи максимально адаптованої до вимог провінції Шаньдун та ринкових умов.

Пояснювальна записка складається зі вступу, п’яти розділів, висновку, списку використаних джерел, містить 50 сторінок, 16 рисунків, 8 таблиць, та 3 додатків.

Ключові слова: база даних, показники інноваційної діяльності, таблиці, автоматизована система, документообіг, імпорт.

ANNOTATION

The paper considers theoretical and practical aspects of software product development for the formation of a database of indicators of innovation in Shandong Province.

The aim of the work is to develop an automated system that is best adapted to the requirements of Shandong Province and market conditions.

The explanatory note consists of an introduction, five chapters, a conclusion, a list of sources used, contains 51 pages, 16 figures, 8 tables, and 3 appendices.

Key words: database, indicators of innovative activity, tables, automated system, document circulation, import.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних.

ЕОМ – електронна обчислювальна машина.

СУБД – система управління базами даних.

ПЗ – програмне забезпечення.

Excel – табличний процесор, програма для роботи з електронними таблицями, створена корпорацією Microsoft для Microsoft Windows, Windows NT і Mac OS. Програма входить до складу офісного пакета Microsoft Office.

.xls – формат електронної таблиці, яка була створена за допомогою системної утиліти Microsoft Excel

XML – eXtensible Markup Language (розширювана мова розмітки).

B/S – Browser/Server архітектура взаємодії між веб-браузером та сервером у якій основна логіка транзакцій виконується на стороні сервера

БЗ – База Знань

СХ – Сховище даних

JSON – JavaScript Object Notation (об’єктний запис JavaScript).

CPU – Центральний процесор

HDD – Hard Disk Drive накопичувач запам’ятовуючого пристрою

ОС – операційна система

UX – User experience (користувацький досвід).

ВСТУП

Будь-яка установа регулюється і залежить від конкретного набору відповідних документів, які покликані структурувати, контролювати та прискорювати усі її робочі процеси. Для провінції Шаньдун, зокрема, для показників інноваційної діяльності важливою складовою документації – є статистичі документи, які відображають основні економічні показники діяльності провінції, а також можуть використовуватися при складанні статистичної звітності. Тому дані отримані із документів статистичної звітності впливають на подальший перебіг процесів як підприємств, так і всієї провінції Шаньдун.

Даний документ створений як технічна специфікація по розробці програмного забезпечення в рамках проекту «Інструментальні засоби формування бази даних показників інноваційної діяльності провінції Шаньдун». У документі описані основні вимоги до проектування системи, закладені основи для створення «технічного завдання на проектування системи». Мета документа - формалізація вимог до функціональності і проектування. Справжня технічна специфікація є підставою для конкретних рішень при проектуванні системи, а також підставою для виконання робіт зі створення програмного забезпечення. Крім цього, даний документ є довідковим матеріалом для конструкторів, тестувальників, адміністраторів і користувачів системи, які можуть більше дізнатися про систему за допомогою цієї специфікації. Дана специфікація призначена для керівництва проектом, системних аналітиків, керівників випробувань системи, керівника команди розробників, конструкторів системи.

Використання баз даних для зберігання інформації допомагає її структурувати, запобігаючи розбіжностям і помилкам, викликаним людським фактором. Хоча частина програми забезпечує точне та чітке

дотримання правил і валідації документів, що підлягають заповненню, для точного обчислення даних необхідні подальші розрахунки. Користувацький інтерфейс полегшує редагування інформації, що міститься в базі даних без використання відповідних інструментів бази даних, роблячи дані більш наочними та інтуїтивно зрозумілими. Незважаючи на значну кількість систем формування баз даних, до теперішнього часу не створено універсальних систем зберігання, обробки та відображення статистичних даних.

Тому напрям дослідження є актуальним.

1 ЗАДАЧА ФОРМУВАННЯ БАЗИ ДАНИХ ПОКАЗНИКІВ ІННОВАЦІЙНОЇ ДІЯЛЬНОСТІ ПРОВІНЦІЇ ШАНЬДУН

Метою роботи є розробка програмного продукту, який створює таблиці показників інноваційної діяльності провінції Шаньдун за посиланням на сервер бази даних, що візуалізовано у вхідних даних листів Excel формату .xls

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- проаналізувати вже існуючі програмні рішення;
- проаналізувати літературні джерела на відповідну тематику;
- обрати засоби та інструменти розробки програмного продукту;
- обрати методи розробки;
- розробити архітектуру програмного забезпечення;
- провести моделюючий експеримент;
- здійснити програмну реалізацію формування бази даних і таблиць з використанням обраних технологій програмування.

Програмна система має включати наступний функціонал:

- створення бази даних на сервері
- введення експериментальних даних, які будуть використовуватись для відображення;
- створення таблиць і заповнення їх даними показників інноваційної діяльності провінції Шаньдун за заданим форматуванням

Вхідними даними програмної системи є:

- індикатори інноваційного розвитку (витрати на науково-дослідну діяльність, кількість патентів, тощо) за період від 5 до 10 років

Вихідними даними є:

- корисні дані на сервері бази даних щодо рівня інноваційного розвитку провінції і його впливу на економічний стан провінції. Індикатори і показники інноваційного розвитку для подальшого прогнозування економічного стану
- таблиці за корисними даними на сервері бази даних та їх відповідне відображення на стороні веб-клієнта

Основними користувачами системи є:

- керівники наукових організацій
- співробітники управлінських органів наукових організацій
- співробітники компетентних органів науки на рівні міста, провінції, держави.

Система повинна бути розроблена за архітектурою B/S, після виконання розробки користувачі повинні бути підключені до сервера. Доступ користувачів до системи реалізується через стандартний веб-браузер.

2 ОГЛЯД ІСНУЮЧИХ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ФОРМУВАННЯ БАЗИ ДАНИХ

2.1 Опис формування бази даних показників інноваційної діяльності провінції Шаньдун

Однією з головних задач даної дипломної роботи є розробка бази даних для забезпечення стабільної та коректної роботи серверної частини за рахунок надання актуальної інформації для генерації відповідних статистичних показників інноваційної діяльності провінції Шаньдун.

Як вже було зазначено, на основі того, що більшість показників є однорідними, або зовсім однаковими даними, тому постає доцільність створення єдиної бази даних що дозволить забезпечувати своїм наповненням всю необхідну інформацію для генерації повного спектру статистичних показників інноваційної діяльності провінції Шаньдун.

Спроектоване рішення дозволить виключити можливість появи помилок при наповненні бази даних, а також допоможе спростити задачу створення відповідного інтерфейсу користувача для редагування та оперування наявними даними.

Таким чином, на першому етапі постає задача вибору типу бази даних, що буде найбільш доцільним для вирішення поставленої задачі, а також відповідного інструментарію для розробки БД, які будуть відповідати наступним вимогам:

- простота та зручність у використанні;
- наявність необхідної кількості матеріалів для побудови статистичних показників;
- підтримка у відкритому доступі.

Наступним етапом є проектування, а також реалізація бази даних, що буде використовуватися серверною частиною системи для зберігання даних. До того ж мають бути враховані усі особливості показників, що розглядаються, та будуть генеруватися, а також повинен бути закладено можливість подальшого розширення структури БД в умовах появи необхідності доповнення системи новим функціоналом.

2.2 Опис програмного продукту Caspio Bridge

Програмний продукт Caspio Bridge – це додаток, що дозволяє зберігати, маніпулювати і публікувати дані до бази даних, без знань програмування і web-дизайну. Під публікацією, розуміється автоматична генерація HTML сторінок для пошуку / перегляду / редагування даних зберігаються в БД. Додатки даної категорії, засновані на моделях «програмне забезпечення як сервіс» та «програмне забезпечення як послуга». Для роботи потрібен тільки сучасний браузер.

Переваги: високий рівень безпеки, гнучкість налаштувань.

Недоліки: висока ціна, відсутність адаптації до вимог законодавства провінції Шаньдун, відсутній експорт з листів xls.

2.3 Опис програмного продукту MyTaskHelper

Програмний продукт MyTaskHelper це онлайн конструктор веб-форм і баз даних. Вам не потрібно буде витрачати час на вивчення будь-якої мови програмування. Створюйте форми, діаграми, карти всього в кілька кліків і помістіть їх на свою веб-сторінку. Функціонал програмного модулю включає в себе:

- створити статичні і динамічні веб форми будь-якої складності;
- налаштувати їх і вбудувати в сайти;
- згенерувати онлайн бази даних;
- імпортувати / експортувати готову БД з файлів / в файли формату Excel або CSV;
- налаштувати БД і вбудувати її в сайт або використовувати в особистих / корпоративних цілях;
- застосовуючи функції сортування даних, пошуку, угруповання і фільтрації створити віджети;

Переваги: зручність, паралельна робота.
Недоліки: користувач, який має доступ до програми, може випадково або навмисно внести в неї зміни, які можуть порушити роботу програми, немає валідації корисних даних на сервері.

2.3 Опис програмного продукту MS ACCESS 2010

Програмне забезпечення Access 2010 і служби Access - компонент SharePoint -можна використовувати для створення додатків веб-бази даних. Це програмне забезпечення дозволяє:

- організувати доступ до даних і забезпечити його безпеку;
- надавати загальний доступ до корпоративних даних через Інтранет або Інтернет.
- створювати додатки для роботи з базами даних, які не потребують використання Access.

Для роботи з веб-базою даних потрібен обліковий запис користувача. Анонімний доступ не підтримується.

Access 2010 і служби Access (додатковий компонент SharePoint) надають платформу для створення баз даних, які можна використовувати в Інтернеті. Для проектування і публікації веб-бази даних використовується Access 2010 і SharePoint, а користувачі, у яких є обліковий запис SharePoint, працюють з нею через веб-браузер.

При публікації веб-бази даних служби Access створюють сайт SharePoint, на якому вона розміщується. Всі об'єкти і дані бази потрапляють в списки SharePoint на цьому сайті.

Переваги: зручність, швидкість доступу до інформації.

Недоліки: більше не підтримується, немає можливості експорту бази на власний сервер БД.

3 ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ ФОРМУВАННЯ БАЗИ ДАНИХ ПОКАЗНИКІВ ІННОВАЦІЙНОЇ ДІЯЛЬНОСТІ ПРОВІНЦІЇ ШАНЬДУН

3.1. Взаємодія програмних засобів

Під час розробки представленої дипломної роботи використано наступні засоби розробки: мова програмування – C#, середовище розробки програмного продукту – Visual Studio 2019, система управління базами даних - Microsoft SQL Server [6], Microsoft Excel (побудова звітів відбувається шляхом Імпорту даних до бази з файлів даних формату xls), з бази даних деталі показників подаються на веб-клієнт у вигляді об’єкту JSON.

На рисунку 3.1 зображена схема взаємодії застосованих технологій.

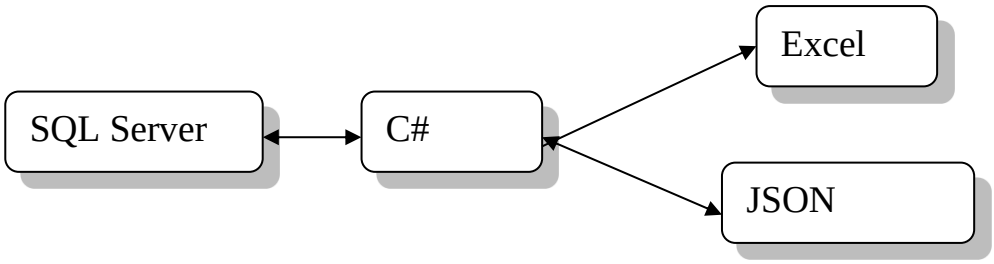


Рисунок 3.1 – Схема взаємодії програмних засобів

3.2. Обґрунтування вибору мови програмування

Для обґрунтування вибору мови програмування для реалізації даного проекту проведемо порівняльний аналіз найбільш розповсюджених середовищ розробки. Результати порівняння мов програмування високого рівня наведено у таблиці 3.1. [7].

Таблиця 3.1. Порівняння мов програмування високого рівня

Назва мови	C++	Visual Basic	C#	Delphi
Швидкість розробки програми	3	4	5	4
Висока продуктивність розробленої програми	5	2	4	4
Низькі вимоги розробленого додатка до ресурсів комп'ютера	3	3	3	5
Можливість розробки нових компонентів і інструментів власними силами	5	3	5	5

У якості основної мови програмування було обрано C# базуючись на таких перевагах:

- велика бібліотека класів;

- велика кількість безкоштовних компонентів, що реалізують необхідні обчислювальні алгоритми, що істотно знижує трудовитрати на реалізацію програми [8];
- простота роботи з пакетом Microsoft Office;
- зручний багатовіконний інтерфейс, який спрощує процес розробки програм;
- підтримка роботи з базами даних;
- сумісність з усіма операційними системами Windows.

C# має широкий набір можливостей. Ця мова усуває необхідність програмувати такі компоненти Windows загального призначення, як мітки, піктограми і діалогові панелі. Діалогові панелі (наприклад, Open File Dialog і Save File Dialog) – компоненти, які часто використовуються, вбудовані безпосередньо в Microsoft Visual Studio C#, які дозволяють пристосувати ці компоненти, щоб вони працювали саме так, як потрібно створюваній програмі [9]. Також тут є заздалегідь створені об'єкти, включаючи кнопки, об'єкти з даними (DataGridView), меню і вже побудовані діалогові панелі. За допомогою цих об'єктів можна, наприклад, забезпечити контроль над даними, які вводить користувач.

3.3. Обґрунтування вибору системи керування базою даних

Результати порівняльного аналізу СКБД наведено у таблиці 3.2. [10]. У якості СУБД для реалізації проекту було обрано Microsoft SQL Server 2015.

Microsoft SQL Server 2015 має надійну, продуктивну та оснащену комплексними засобами аналізу платформу, здатну підтримувати критично важливі прикладні програми, які висувають високі вимоги до ресурсів [11]. Використання цієї платформи дає змогу скоротити час розробки прикладної програми, зменшити витрати часу на управління [12].

Враховуючи вищезазначені критерії, можна сказати, що вибрана СКБД MS SQL Server, яка використовувалась в розробці програмного продукту, є раціональним вибором серед ряду інших.

Таблиця 3.2. Порівняльна характеристика існуючих СКБД

Назва продукту	Основні переваги
Microsoft SQL Server	Високий ступінь захисту даних. Потужні засоби роботи з даними. Висока продуктивність.
Microsoft Access	Простота освоєння та використання. Має потужні засоби підготовки звітів з БД різних форматів.
Microsoft Visual FoxPro	Високий рівень об'єктної моделі. Висока швидкість обробки даних. Інтеграція об'єктно-орієнтованої мови програмування з Xbase і SQL.

Сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування (за стандартом ISO/IEC 2382:2015) та відповідно до специфікацій ПЗ що встановлюється на ЕОМ називається базою даних [1]. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином,

сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

В загальному випадку базою даних можна вважати будь-який впорядкований набір даних. Наприклад, паперову картотеку з формулярами про працівників підприємства у відділі кадрів. В даному випадку ми розглядаємо використання баз даних в інформаційних. На даний час застосунки для роботи з базами даних є одними з найпоширеніших прикладних програм.

Існують централізовані та розподілені бази даних – класифікація за принципом розподіленості. Централізовані – повністю підтримуються на одній ЕОМ. розподіленими вважаються такі, компоненти яких розміщуються в різних вузлах комп'ютерної мережі відповідно до певного критерію. Очевидно, що розподілені БД є більш надійними, оскільки при відмові одного з вузлів в мережі інші залишаються дієздатними, а за умови, що кожен вузол буде мати резервні копії даних з інших вузлів, система буде мати високий рівень відмовостійкості. Однак, такі БД вимагатиме значно більших ресурсів для підтримки роботи, а в контексті обробки даних для формування показників інноваційної діяльності провінції Шаньдун, потрібно враховувати обмеженість ресурсів, як людських, так і матеріальних.

Також при взаємодії мережевих компонентів можна очікувати деградацію швидкодії в порівнянні з централізованою системою. Для вирішення даного завдання ми обираємо централізований тип БД, що значно спрощує підтримку, використання та прискорює операції з даними у БД. Інсталяцію та обслуговування такої БД може бути виконано на сервері установи, або безпосередньо встановлена на ЕОМ працівника, відповідального за оперування та формування показників інноваційної діяльності провінції Шаньдун, що генеруються розроблюваною автоматизованою системою, надійність бази даних та збереження поточного

стану можна забезпечити регулярним створенням резервних копій бази даних – дампів БД.

Бази даних також класифікуються залежно від середовища постійного зберігання даних: у вторинній, тобто постійній пам'яті комп'ютера (як правило, на жорсткому диску), оперативній пам'яті або у третинній - знімних носіях. В даному випадку, оскільки для підтримки працездатності системи одним із важливих факторів є надійність збереження даних та їх послідовність. Тому оптимальним варіантом є традиційна модель зберігання даних – у вторинній пам’яті. Операції з БД, що зберігається у вторинній пам’яті, поступаються рівнем швидкодії БД, що зберігаються в оперативній пам’яті, проте є менш вибагливою до характеристик ЕОМ, на якій буде обслуговуватися БД.

Ще одна основна класифікація баз даних - класифікація за моделлю даних. Модель даних визначає логічну структуру баз даних та визначає основні принципи того, як можна зберігати, організовувати дані та як працювати з ними [2].

У ієрархічній моделі БД дані подаються у вигляді графу-дерева. Вони зберігаються як записи, що поєднуються між собою за допомогою посилань (зв’язків) (Рис. 3.2). Запис – це набір полів, кожне з яких містить лише одне значення. Тип сутності запису визначає, поля якого типу даних запис має.

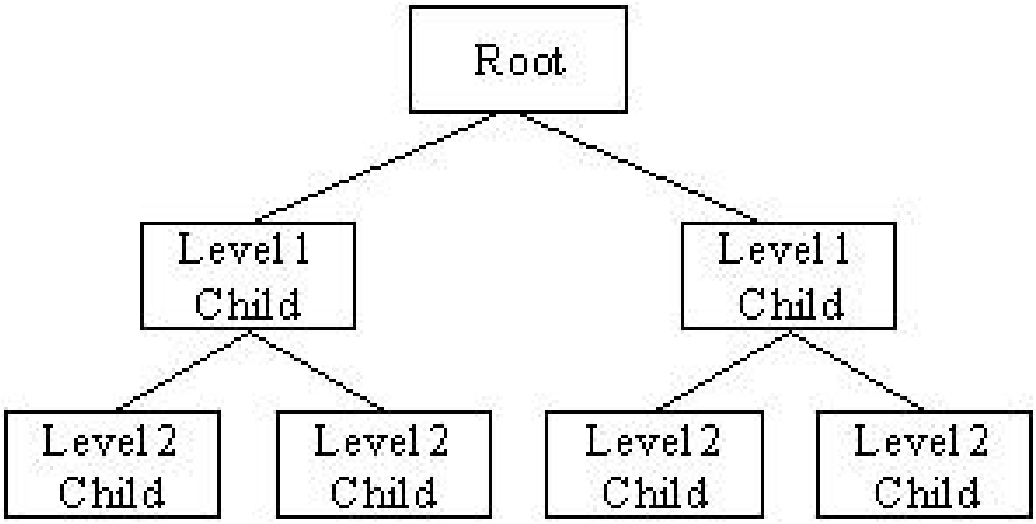


Рисунок 3.2 – Ієрархічна модель даних БД

Записом у ієрархічній моделі БД є рядок (або кортеж) у реляційні моделі. Типом сутності є таблиця у БД.

Ієрархічна модель чітко визначає, що кожен запис нащадків має лише один запис предків, тоді як один предок може мати декілька нащадків. Для вилучення даних з ієрархічної моделі все дерево повинно бути обведене від головного вузла. Ця модель, визнана першою моделлю бази даних, була створена IBM в 1960-х роках, тому багато фахівців вважають її застарілою [3].

Реляційна модель даних використовує підхід обробки даних, використовуючи структуру та мову, сумісну з логікою предикатів першого порядку, вперше описаною Едгаром Коддом, в якій всі дані представлені у вигляді кортежів, згрупованих у взаємозв'язки. Бази даних, організовані в рамках цієї моделі, називаються реляційними.

Пояснення відносин між сутностями БД зображено на рис. 3.3.

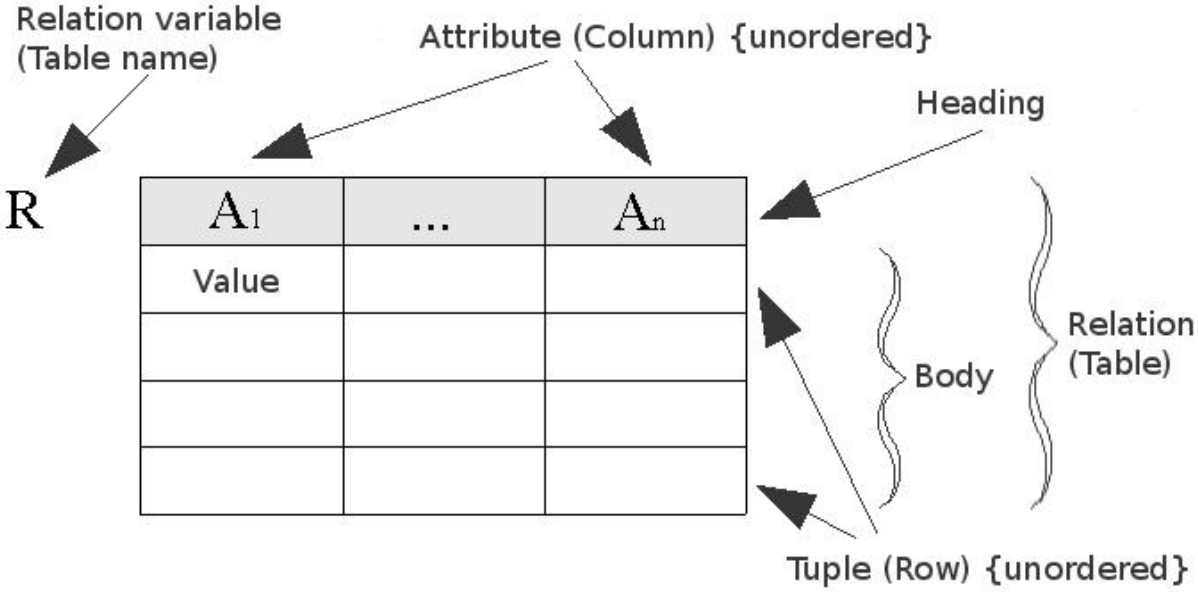


Рисунок 3.3 – Реляційна модель БД та відносин

Мета реляційної моделі полягає у наданні декларативного методу визначення даних та запитів: користувачі чітко вказують, яку інформацію містить база даних і що вони хочуть отримати, зберігаючи всі деталі реалізації програмного забезпечення бази даних. Більшість баз даних використовують мову SQL. Він стандартизований, але різні реалізації в певних СУБД можуть дещо відрізнятися від цього стандарту, щоб забезпечити більше можливостей або підвищити продуктивність.

Слід зазначити, що ця модель організації даних в базі даних на даний момент є найбільш поширеною і має безліч реалізацій, як з відкритим програмним кодом, так і комерційних платних систем.

У об'єктно-орієнтованій моделі інформація представлена як об'єкти, як і в об'єктно-орієнтованих мовах програмування. Це значно спрощує взаємодію між базою даних та ОО - мовами програмування, немає необхідності використовувати ORM фреймворки або писати об'єкти для доступу до даних БД.

Для використання в цій роботі було вирішено обрати реляційну базу даних, оскільки цей формат є широко поширений та є багато джерел інформації про використання таких баз даних, а також велика кількість баз даних з відкритим вихідним кодом або таких що є безкоштовними для некомерційного використання, тобто можна знайти оптимальне програмне забезпечення для використання в розробленій системі.

На сьогоднішній день існує багато систем управління базами даних. Розглянемо кілька з них:

СУБД PostgreSQL - це безкоштовна об'єктно-реляційна система управління базами даних. Це одна з найсучасніших СУБД, головним чином орієнтована на повне дотримання стандартів та розширення, тобто вона повинна повністю відповідати стандартам ANSI / ISO SQL. PostgreSQL також підкреслює, серед іншого, що він має об'єктно-орієнтовану функціональність і підтримує відповідні концепції.

Система PostgreSQL базується на ядрі, що створюється великою кількістю розробників. У подібних випадках необхідно зосереджуватись на оснащенні системи новими можливостями, замість оптимізації існуючих рішень, оскільки за необхідності завжди є можливість повернутися до оптимізації відповідних ділянок коду.

Також необхідно зазначити, що система PostgreSQL недостатньо оптимізована для вирішення повсякденних не дуже важких задач. Використання системи обґрунтовано у випадках, коли необхідна підвищена надійність та підтримка об’єктних підходів до бази даних. У випадку, коли першочерговою задачею постає виконання простих операцій зчитування або запису, PostgreSQL показує не найкращі результати [4].

Oracle Database - це об'єктна система управління реляційними базами даних від Oracle Corporation. Ця база даних забезпечує ефективне, надійне та безпечне управління даними критично важливих програм, таких як онлайн-середовища, виконує масштабну обробку транзакцій (OLTP), сховища даних з високим потоком та ресурсомісткі Інтернет-програми. Oracle Database Enterprise Edition надає інструменти та функції, які відповідають доступності та масштабованості сучасних бізнес-додатків. Це видання включає всі компоненти Oracle Database, а також дозволяє розширити, придбавши додаткові модулі та програми [5].

Система Oracle Database дозволяє звертатися до даних з будь-якого джерела – клієнту БД або додатку, розробленого із застосуванням технологій MS .NET, Visual Studio та веб-додатків. Головною умовою є лише наявність бібліотек, що дозволяють підключатися до серверу бази даних Oracle. Система Oracle Database є комерційною СУБД, проте є безкоштовна версія, яку можна скачати з офіційного сайту компанії Oracle, але безкоштовна версія має зменшений функціонал.

Microsoft SQL Server заснований на концепції платформи даних Microsoft: це спрощує управління даними в будь-який час і в будь-якому

місці. Microsoft SQL Server дозволяє зберігати в базах даних інформацію, отриману із структурованих, напівструктурованих та неструктурованих джерел, таких як зображення та музика. У SQL Server є безліч інтегрованих сервісів, які розширюють можливості використання даних: ви можете запускати запити, виконувати пошук, синхронізувати, звітувати та аналізувати дані. Усі дані зберігаються на основних серверах, що входять до центру обробки даних. Вони доступні з настільних та мобільних пристроїв. Таким чином ви зможете повністю керувати даними, незалежно від того, де вони зберігаються.

За допомогою MS SQL Server ви можете отримати доступ до даних із будь-якого додатка, розробленого за допомогою технологій MS .NET та Visual Studio, а також в сервісно-орієнтованій архітектурі та бізнес-процесах через Microsoft BizTalk Server. SQL Server дозволяє створити надійну, продуктивну та інтелектуальну платформу, яка відповідає всім вимогам для роботи з даними [6]. Ця система є комерційною.

Безкоштовна система управління базами даних. MySQL належить корпорації Oracle, яку вона придбала разом із придбаною Sun Microsystems, яка розробляє та підтримує програму. Поширюється за загальною публічною ліцензією GNU або за власною комерційною ліцензією. Крім того, розробники створюють функціональність для замовлення ліцензованих користувачів.

MySQL - це рішення для малих та середніх програм. Він є частиною серверів WAMP, LAMP і в портативних збірках серверів Denver, XAMPP. Зазвичай MySQL використовується як сервер, до якого звертаються місцеві або віддалені клієнти, але дистрибутив містить внутрішню бібліотеку серверів, яка дозволяє включати MySQL в автономні програми [7].

Як було сказано раніше, обрані СУБД повинні відповідати вимогам щодо простоти використання та наявності достатнього довідкового матеріалу. Тому, було обрано MS SQL - дана СУБД дозволяє реалізувати

увесь необхідний функціонал для операційної бази данних для формування показників інноваційної діяльності провінції Шаньдун, а також дана СУБД є однією з найбільш розповсюджених, саме тому в мережі є велика кількість матеріалів, як довідкових, включаючи офіційний сайт, так і навчальних, де розглядаються найрізноманітніші проблеми та шляхи їх вирішення.

3.4 Середовище розробки програмного продукту

Microsoft Visual Studio 2019 має багато переваг у порівнянні з іншими середовищами. Порівняльну характеристику наведено у таблиці 3.3 [13].

Таблиця 3.3 - Порівняльний аналіз існуючих середовищ розробки ПЗ

Назва / Параметри	Turbo C# Explorer	Borland C#Builder	Microsoft Visual Studio
Версія	Turbo C# Explorer 7	Borland C#Builder 6	Microsoft Visual Studio 2019
Фірма виробник	Expl	Borland	Microsoft Corporation
ОС	Windows XP и выше	Windows XP и выше	Windows XP и выше
Підхід до розробки	Об’єктно – орієнтований	Об’єктно – орієнтований	Об’єктно – орієнтований
Утиліти для роботи з БД	- Database Desktop; - BDE Administrator; - SQL Explorer; - SQL Monitor	- Database Desktop; - BDE Administrator; - SQL Explorer; - SQL Monitor	- Solution Explorer; - Server Explorer; - Data Base Diagram Designer; - Table Designer; - Query and View Designer
Підтримка стандарту мови SQL	+	+	+

Наявність компонент побудови звітів	Rave Reports Borland Editions	Quick Reports	- Crystal Reports Windows Forms Viewer; - Crystal Reports Engine
Підтримка Windows-подібного інтерфейсу	+	+	+
Можливість створення *.exe	+	+	+

Таким чином для аналізу були відібрані засоби розробки: Turbo C# Explorer 7, Borland C# Builder 6, Microsoft Visual Studio 2019, як найбільш популярні. Всі три засоби мають великі можливості для створення додатків, що взаємодіють з базами даних [14]. І все ж Microsoft Visual Studio 2019 є найбільш потужним інструментом для розробки програм. Саме його вирізняють такі особливості як:

- можливість працювати з обраною СКБД SQL Server прямо з середовища розробки;
- можливість працювати з усіма COM – об’єктами Windows [15];
- створення програмного продукту з використанням останніх можливостей .NET Core [16];
- швидкість та зручність розробки.

Наводиться перелік переваг MS SQL [9] для більш чіткого обґрунтування вибору СУБД:

- Продуманість та швидкодія.
- Займає небагато дискового простору.
- Може бути легко встановлена на базі багатьох операційних систем,

таких як Unix-like, Windows, тощо.

Враховуючи поширення в Інтернеті, є досить обширна база знань що допоможе знайти багато матеріалів, які вирішують певні питання та проблеми, що виникають при роботі з нею. Добре підходить для розробки невеликих застосунків із великими об’ємами даних.

В результаті аналізу, було обґрунтовано обрано Visual Studio 2019, як середовище для розробки системи формування бази даних показників інноваційної діяльності провінції Шаньдун.

Для розробки програмного продукту було обрано:

- СКБД SQL Server 2015;
- Середовище розробки Visual Studio 2019;
- Мова програмування C#.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Концептуальна схема системи

Основна мета створення будь-якої програмної системи - створення програмного продукту, який допомагає користувачеві виконувати операції із програмою за допомогою зручного графічного інтерфейсу та UX. Для створення таких програм, насамперед, визначаються вимоги, яким повинна задовольняти система та вимоги до програмного і графічного інтерфейсу системи.

Для полегшення процесу розробки системи було визначено функціональні вимоги до системи [7]. Ця інформація подана у вигляді діаграми Use Case, за якою була створена система (Рис 4.1).

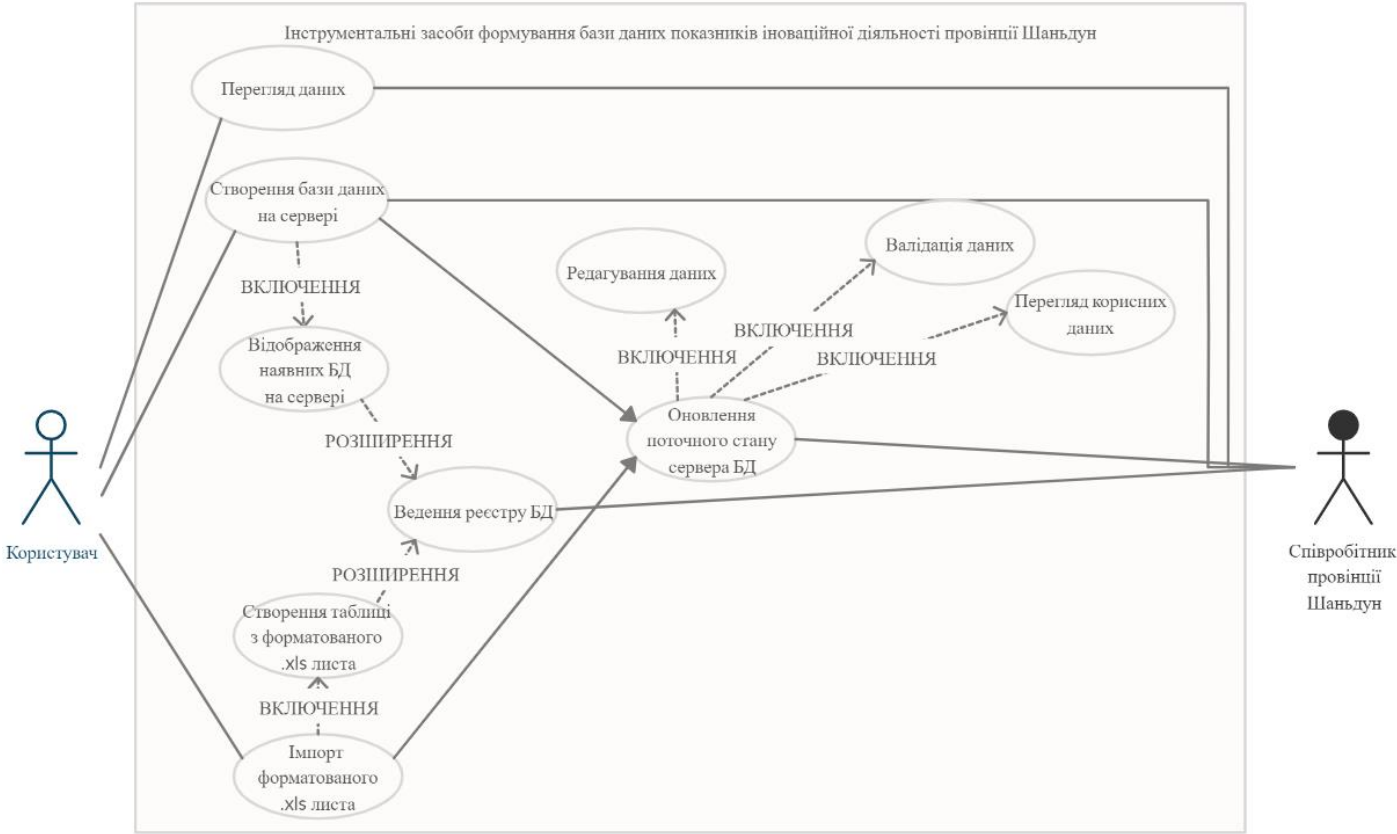


Рисунок 4.1 - Діаграма Use Case

4.2 Архітектура системи

Проект «Інструментальні засоби формування бази даних показників інноваційної діяльності провінції Шаньдун виконано на основі B/S архітектури і трирівневої моделі сервісів системи (Рис. 4.2)

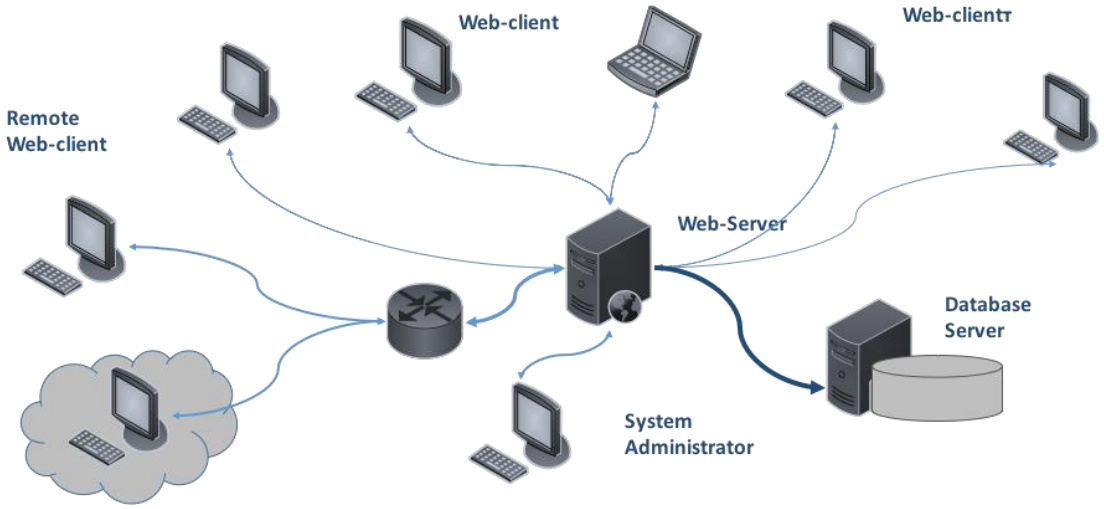


Рисунок 4.2 – Структурна схема системи

Системна архітектура представляє собою загальну логічну організацію системи, що визначає процес її функціонування, включаючи методи обробки даних, склад, фізичну структуру складових компонент, принципи взаємодії віддалених користувачів з серверами і сховищами даних через мережі передачі даних з урахуванням їх топології і територіального устрою провінції Шаньдун.

Основу системної архітектури становить трирівнева архітектура: сервер БД, сервер додатків, Web-клієнт. Користувачі можуть бути об'єднані в мережу і взаємодіють з БД, БЗ, СД в процесі вирішення функціональних завдань.

Для забезпечення інформаційної взаємодії з різними джерелами і споживачами інформації архітектура системи базується на використанні

стандартних протоколів, інтерфейсів сполучень і комунікаційних засобів стандарту OSI (взаємодія відкритих систем), має можливості для розширення переліку функцій, підключення додаткових користувачів, Рис 4.3.

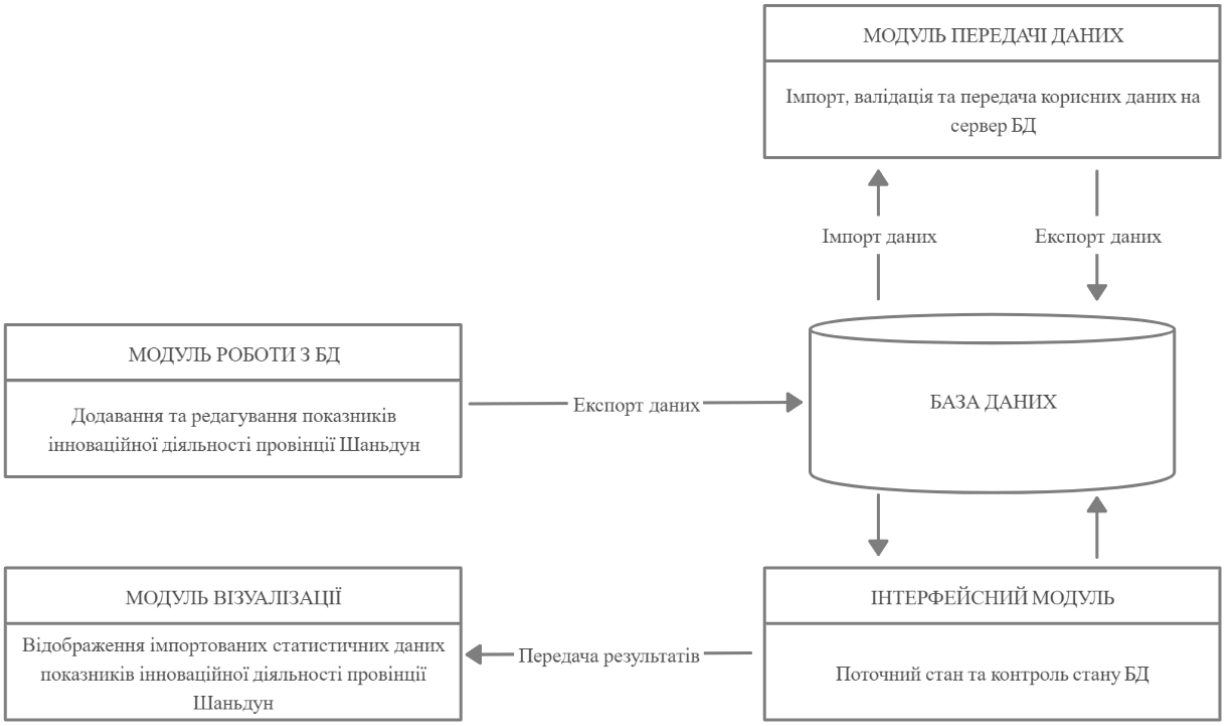


Рисунок 4.3 Архітектура взаємодії системи

До складу архітектури системи входять такі основні компоненти:

1. Сервер баз даних для зберігання і здійснення доступу до даних. Сервер повинен забезпечувати можливістю «гарячої» заміни дисків з метою підвищення його надійності.
2. Сервер додатків для вирішення функціональних комплексів задач аналітичної обробки даних;
3. Web-клієнт аналітика - користувача системи (керівники всіх рівнів державних відомств провінції Шаньдун);
4. Web-клієнт з інтерфейсом адміністратора даних для введення необмеженої кількості нових показників;

5. Інформаційна база системи забезпечує накопичення, зберігання і доступ до інформації.
6. Загальносистемне, мережеве та прикладне програмне забезпечення.
7. Засоби введення, виведення, відображення і документування інформації.

4.2 Алгоритми роботи системи

Створений програмний комплекс відповідає структурному шаблону MVC, усі підпрограми оформлено в окремому модулі – контролері, а абстрактні класи об’єктів у моделі-відображенні. Алгоритм отримання імен баз даних виконується за допомогою запиту $"SELECT * FROM {model.DbName}.dbo.{model.TblName}" який повертає імена баз даних на сервері MS SQL, імена зберігаються у JSON об’єкт та можуть бути використані для доступу до наявних БД у подальщому.

При перегляді даних на інтерфейсі користувача реалізовано алгоритм зв’язаного списку за відповідністю База Даних – Імена наявних таблиць, ім’я баз даних що не мають таблиць для перегляду вилучаються з даного списку. Так як наявні таблиці на сервері мають лише корисні дані – доповнення таких таблиць валідується на стороні сервера.

Реалізовано обробку помилок та повернення повідомлень від БД або сервера додатку на інтерфейс користувача, прапор IsOk є індикатором коректності будь яких операцій виконуваних на інтерфейсі користувача (веб-додатку).

Основний алгоритм роботи системи, а саме алгоритм імпорту форматованого xls листа до бази даних, представлений на Рис. 4.4

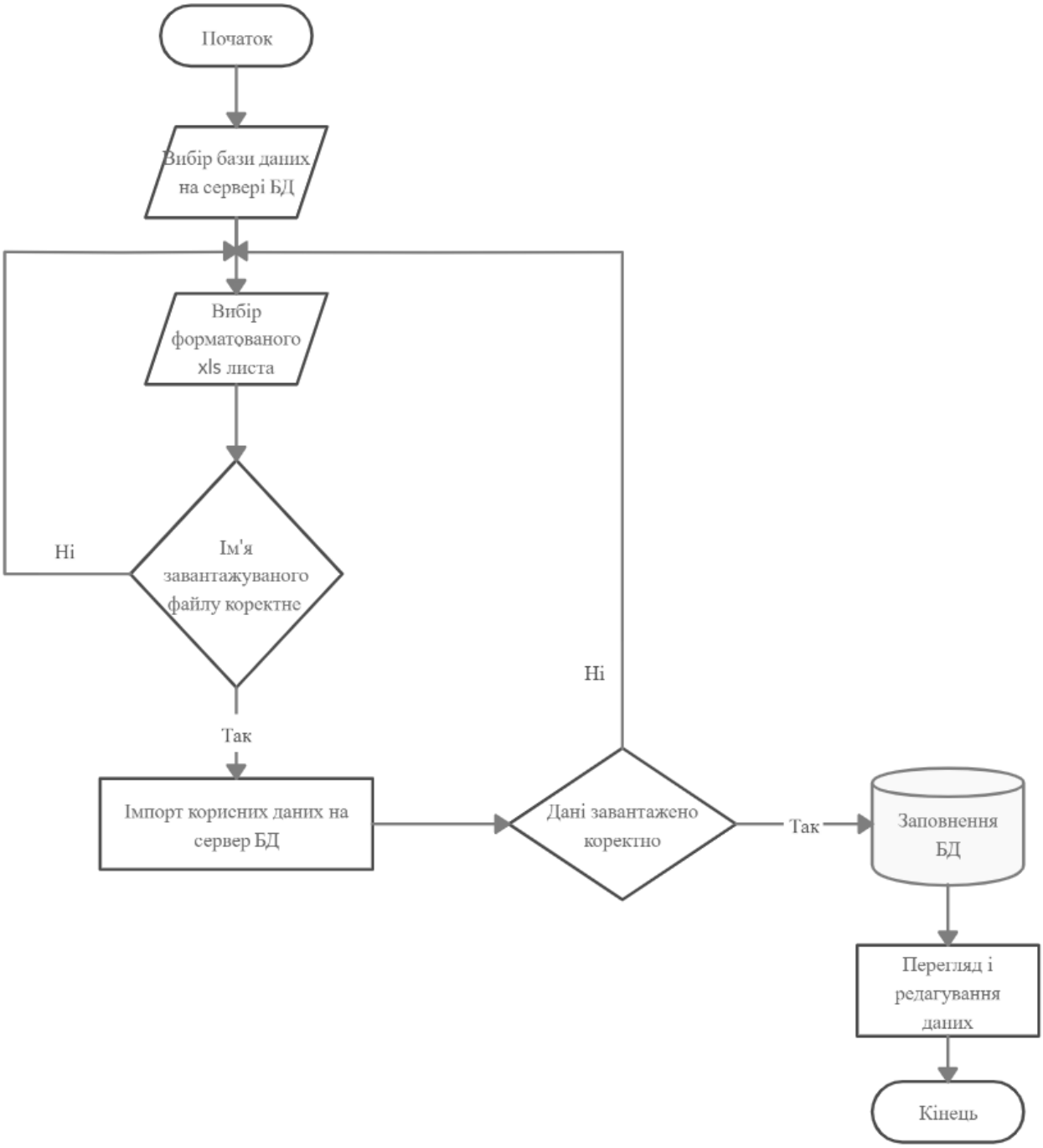


Рисунок 4.4 Алгоритм імпорту форматованого xls листа

4.3 Опис бази даних

Для реалізації поставленої задачі було спроектовано базу даних, що динамічно створює таблиці в залежності від типів даних та назв колонок у вхідному файлі. Схема шести створених таблиць та типів даних наведено на Рис. 4.5:

data1818 *

Column Name	Condensed Type
Id	int
Категорія	varchar(255)
[Кількість персоналу, дослідження та розробка]	decimal(18, 3)
[Базові дослідження (персонал)]	decimal(18, 3)
[Прикладні дослідження (персонал)]	decimal(18, 3)
[Експериментальні дослідження (персонал)]	decimal(18, 3)

data_1822 *

Column Name	Condensed Type
Id	int
Категорія	varchar(255)
[Розробка та дослідження (персонал)]	decimal(18, 3)
[На проекті цього року (персонал)]	decimal(18, 3)
[Менеджмент та послуги (персонал)]	decimal(18, 3)
[Повний робочий день (персонал)]	decimal(18, 3)
[Неповний робочий день (персонал)]	decimal(18, 3)

data_1820 *

Column Name	Condensed Type
Id	int
Категорія	varchar(255)
[Кількість розробок та досліджень (ю...	decimal(18, 3)
[Персонал досліджень (персонал)]]]	decimal(18, 3)
[Повний робочий день (персонал)]	decimal(18, 3)
[Неповний робочий день (персонал)]	decimal(18, 3)
[Бакалавр (персонал)]	decimal(18, 3)
[Магістр (персонал)]	decimal(18, 3)

data_1827 *

Column Name	Condensed Type
Id	int
Категорія	varchar(255)
[2009]	decimal(18, 3)
[2010]	decimal(18, 3)
[2011]	decimal(18, 3)
[2012]	decimal(18, 3)
[2013]	decimal(18, 3)

data_1819 *

Column Name	Condensed Type
Id	int
Категорія	varchar(255)
[Витрати на розробку і дослідження]	decimal(18, 3)
[Додаткові витрати на розробку і дослідження]	decimal(18, 3)
[Базові дослідження]	decimal(18, 3)
[Прикладні дослідження]	decimal(18, 3)
[Експериментальні дослідження]	decimal(18, 3)
[Державні фонди асигнувань]	decimal(18, 3)
[Самостійно залучені кошти підприємств]	decimal(18, 3)
[Іноземні фонди]	decimal(18, 3)
[Інші Фонди]	decimal(18, 3)
[Вітчизняні науково-дослідні установи]	decimal(18, 3)
[Витрати на державні Коледжі та Університети]	decimal(18, 3)
[Витрати на державні підприємства]	decimal(18, 3)
[Витрати за кордон]	decimal(18, 3)

data_1817 *

Column Name	Condensed Type
Id	int
Категорія	varchar(255)
[Кількість прийнятих заявок на патенти]	decimal(18, 3)
[Кількість наданих заявок на патенти]	decimal(18, 3)
Винаходи	decimal(18, 3)
[Корисні моделі]	decimal(18, 3)
Конструкції	decimal(18, 3)
[Моделі (всього)]	decimal(18, 3)

Рисунок 4.5 – Схема бази даних

5 РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА

5.1 Аналіз можливих реалізацій та вимоги до розроблюваних у роботі програмних та апаратних засобів

Метою розробки графічного інтерфейсу користувача та користувацького досвіду є створення ефективного засобу взаємодії між людиною та системою. Інтерфейс повинен підтримувати всю доступну функціональність на стороні сервера, забезпечувати доступ для перегляду всієї інформації, що зберігається в базі даних, а також можливість редагувати та доповнювати цю інформацію. Крім того, ці властивості повинні бути реалізовані таким чином, щоб використання системи було зручним і зрозумілим.

Графічний інтерфейс розроблено за допомогою бібліотеки React-bootstrap що є бібліотекою з відкритим вихідним кодом.

Доступ до користувацького інтерфейсу забезпечується беспосередньо через веб-браузер, конфігурація клієнта наведена у Таблиці 5.1.

Таблиця 5.1 -- Мінімальна конфігурація клієнта

Найменування	Параметри конфігурації
CPU	Intel Core i3
Пам'ять	≥4G
HDD	Об'єм вільної пам'яті на диску≥10G
Мережева карта	100M/1000M адаптивний мережевий адаптер

Мінімальна конфігурація апаратного забезпечення сервера наведена у Таблиці 5.2

Таблиця 5.2 – Мінімальна конфігурація сервера

Найменування	Параметри конфігурації
CPU	Intel 2.5GHz
Пам'ять	≥8G
HDD	Об'єм вільної пам'яті на диску ≥50G
Мережева карта	100M/1000M адаптивний мережевий адаптер

Конфігурацію програмного забезпечення клієнта наведено у Таблиці 1.3

Таблиця 5.3 -- Мінімальна конфігурація клієнта

Найменування	Параметри конфігурації
ОС	WinXP та вище, UNIX based OS з графічним інтерфейсом
Умови роботи	IE7.0 та вище Google Chrome 60.0 та вище MS Office 2003 та вище, або інші сумісні програмні системи

Конфігурацію програмного забезпечення сервера для роботи системи наведено у Таблиці 5.4

Таблиця 5.4 – Мінімальна конфігурація сервера

Найменування	Параметри конфігурації
ОС	Windows Server 2003 або вище
Умови роботи	MSSQL Server 13.0 .Net Core SDK 3.1

При виборі підходу до розробки та реалізації графічного інтерфейсу програми розглядалися наступні варіанти побудови інтерфейсу:

- крос-платформенний інтерфейс;
- веб-інтерфейс

Крос-платформенний інтерфейс користувача вирішує проблему запуску додатків на різних ОС. В залежності від операційної системи та незалежно від того, як технічно досягнута працездатність цих додатків, - стандартні елементи інтерфейсу мають різні розміри. Саме тому просте жорстке позиціонування елементів інтерфейсу стає неможливим, оскільки під іншою ОС в додатках можуть виникати проблеми з відображеннями (насування елементів один на одного, тощо).

Для вирішення проблеми з досягенням коректного відображення елементів додатку на різних ОС існую декілька підходів. Порівняльний аналіз наведено у таблиці 5.5.

Таблиця 5.5 – Порівняльний аналіз методів досягнення кросплатформенності інтерфейсу

Підхід	Опис	Переваги	Недоліки
1. Єдиний стиль додатку, загальний для всіх ОС.	Програми виглядають однаково незалежно від операційної системи, на базі якої запускаються. Таким чином працюють інтерфейсні бібліотеки Java, наприклад Swing.	1. Є можливість жорско розставляти елементи управління (як наприклад у Delphi); 2. Оригінальний стиль.	1. Системі необхідно мати свої екранні шрифти; 2. Стиль відрізняється від стилю операційної системи.

Таблиця 5.5 (Продовження) – Порівняльний аналіз методів досягнення кросплатформенності інтерфейсу

Підхід	Опис	Переваги	Недоліки
2.Інтерфейс, що самоадаптується	Інтерфейс автоматично підлаштовує сітку під реальні розміри елементів управління. Розповсюджені приклади: Qt, wxWidgets, XUL.	1. Стандартний стиль, що швидко підлаштовуються під стиль ОС;	1. Для налаштування сітки з можливістю самоадаптації необхідні висококваліфіковані фахівці-програмісти; 1. Проблеми з щільною компоновкою.
Гібридний підхід	Реалізовано у GTK+	1. Автоматична локалізація. 2. Можливість використовувати шрифти з системи а не “тягти свої”; 3. Певна автоматизація локалізації.	2. Перейняв усі недоліки попередніх двох методів; 3. Стиль відрізняється від стилю ОС; 4. Часто виникають проблеми з щільним компонуванням.

5.2 Веб-інтерфейс

Веб-інтерфейс -- веб-сторінка або сукупність веб-сторінок, що надає користувальницький інтерфейс для взаємодії з сервісом або пристроєм за допомогою протоколу HTTP і веб-браузера. Веб-інтерфейси набули широкого поширення в зв'язку з ростом популярності всесвітньої павутини [8] і відповідно - бути широко розповсюдженим веб-браузерів.

Одним з основних вимог до веб-інтерфейсів є їх однаковий зовнішній вигляд і однакова функціональність при роботі в різних браузерах.

Веб-інтерфейси виявляються менш вибагливими в процесі розробки додатків, а також не потребують встановленя жодного додаткового ПЗ на ПК.

Оскільки розроблена система побудована за принципами клієнт-серверної архітектури, клієнт виступає у якості окремої підсистеми, яка буде виконувати свої функції за допомогою взаємодії з прикладним програмним інтерфейсом (API), який надаватиме серверна частина застосунку.

В рамках даної роботи розроблена клієнтська частина буде взаємодіяти з сервером застосунку через веб-додаток, реалізований на базі фреймворку React.js. Клієнт взаємодіє з сервером за допомогою протоколу HTTP(S), використовуючи відповідні дані у вигляді JSON документів та протокольні запити. Загальна схема взаємодії представлена на рисунку 5.1.

Незважаючи на те, що серед переваг повноцінної програми є більша гнучкість алгоритмів та операцій з даними, яка доступна при написанні коду, для реалізації даного проекту було все ж обрано варіант веб-

інтерфейсу. Оскільки розробка повноцінної програми буде потребувати набагато більше часу, а також висуватиме вимоги до апаратної платформи. Також, можна створити крос-платформну реалізацію, за допомогою мови програмування Java. В такому разі для використання системи користувачу необхідно мати встановлене середовище виконання відповідної Java (JRE).

Рис. 5.1 – Загальний принцип мережевої клієнт-серверної взаємодії

З іншого боку, веб-інтерфейс дозволяє використовувати систему без встановлення додаткового програмного забезпечення на комп’ютер. Тобто, доступ до системи може здійснюватися через взаємодію між сервером відділу та машиною користувача. По-перше, це зменшує вимоги до апаратних можливостей робочої станції, яким надається доступ, по-друге, програму можна довго змінювати, щоб забезпечити повноцінний багатокористувацький режим. Однак слід зазначити, що на етапі налагодження програми вся система буде встановлена на одному комп’ютері.

З огляду на переваги реалізації веб-інтерфейсу, ця опція вибирається як частина клієнта.

5.3 Реалізація інтерфейсу користувача

Для реалізації клієнтської частини обрано веб-інтерфейс. Це дозволить пришвидшити процес розробки та спростити використання розгортання системи без втрати візуальної якості та зручності у використанні а також надасть можливості кросс-платформенної розробки та доступу до інтерфейсу через веб-браузер.

Веб-інтерфейс користувача було реалізовано засобами клієнтської бібліотеки React що використовує HTML, CSS, JavaScript для взаємодії з користувачем (веб-браузер), це пришвидшило взаємодію із серверною частиною та дозволило вести розробку серверної та клієнтської частини використовуючи одну платформу та оточення розробки. На рис. 5.2 зображено інтерфейс створення бази даних на сервері, вибір бази даних для збереження статистичної інформації (використовується форматований .xls лист), та завантаження корисної інформації на сервер

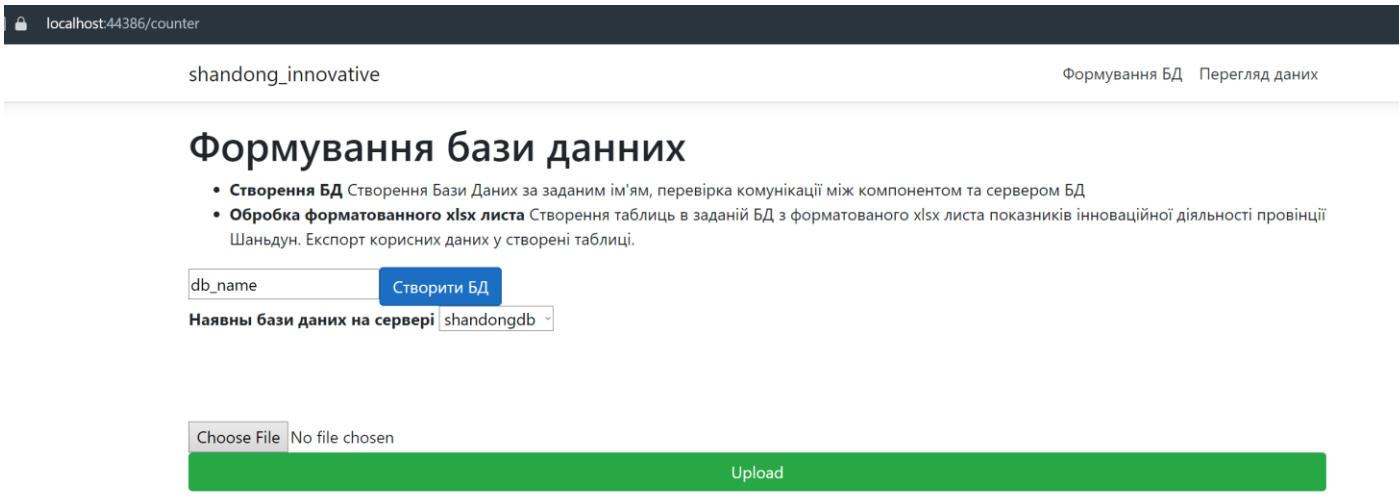


Рисунок 5.2 -- Інтерфейс основної частини програми

Для більшої зручності користування дані було розбито на 3 категорії: «Інфо», «Формування БД», «Перегляд даних».

- Підпункт меню «Формування БД» містить доступ до створення бази даних на сервері та створення таблиці за ім'ям файлу
- Підпункт меню «Перегляд даних» надає доступ до перегляду та редагування інформації, що стосується наявних статистичних даних показників інноваційної діяльності провінції Шаньдун

Як зазначено на рис. 5.3 – на сторінці що відповідає підпункту «Перегляд Даних» реалізовано зв'язаний випадаючий список що в залежності від обраної бази даних відображає усі наявні таблиці які можливо відобразити та редагувати.

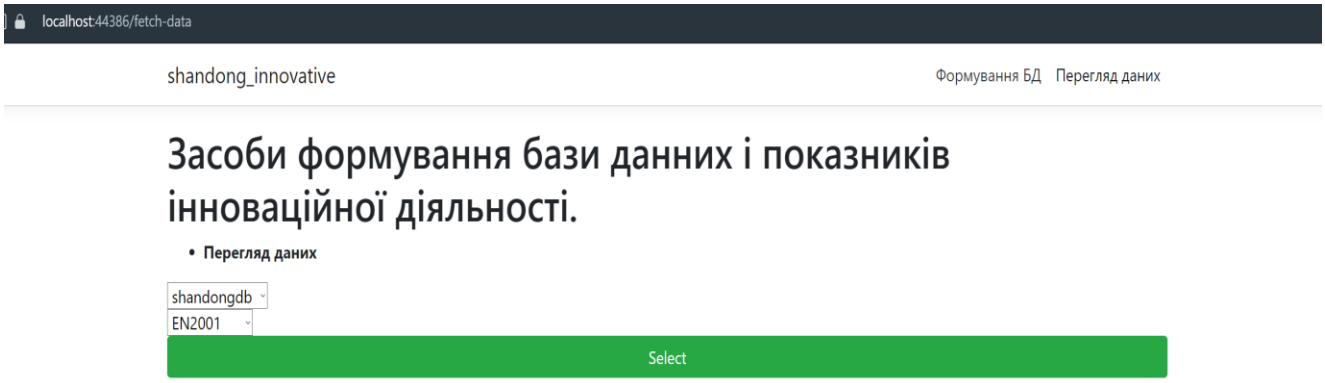
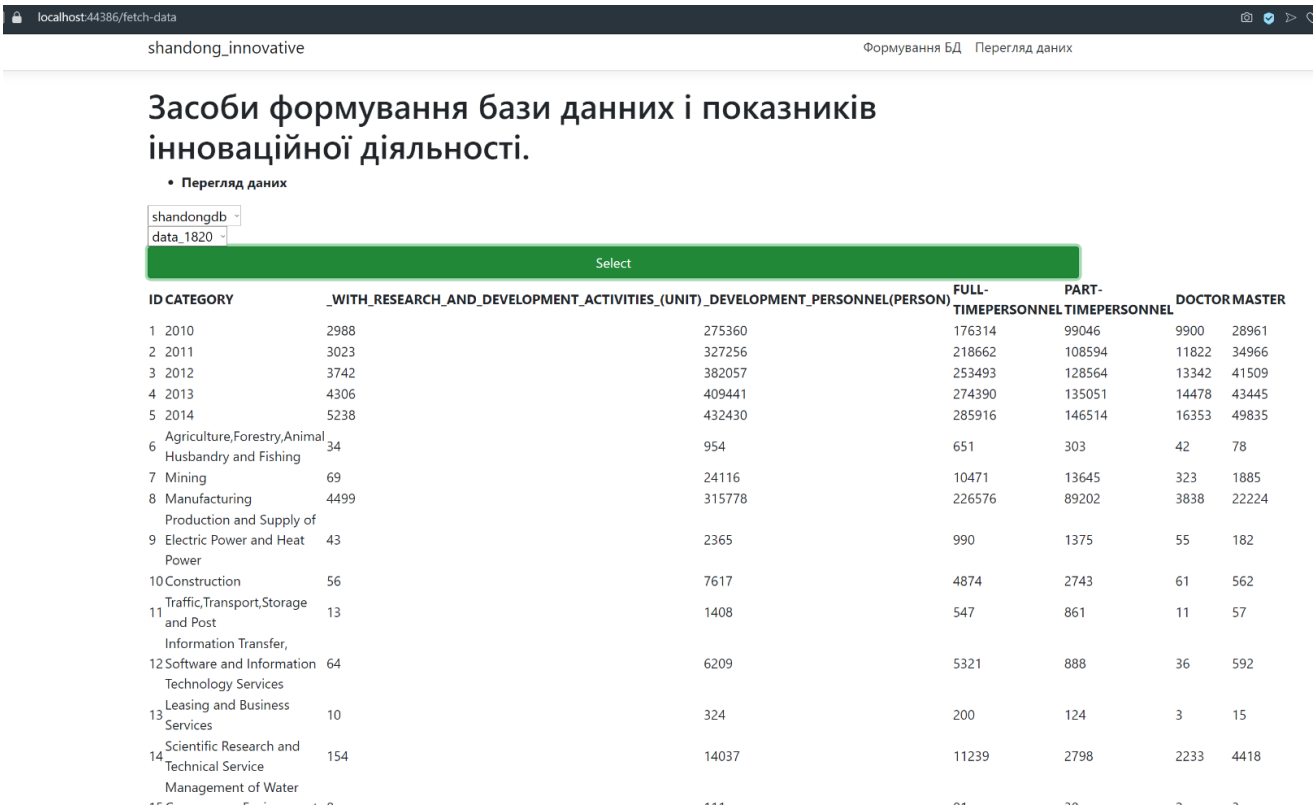


Рисунок 5.3 -- Інтерфейс вибору наявних баз даних для перегляду

Перегляд даних та їх редагування здійснюється у відповідному елементі інтерфейсу як зазначено на рис. 5.4

На даній формі інтерфейсу користувача також відображається ідентифікатор запису в таблиці який є первинним ключем для реляційної

бази даних, таблиця та її межі будуються динамічно в залежності від кількості даних та полів на сервері БД.



ID	CATEGORY	_WITH_RESEARCH_AND_DEVELOPMENT_ACTIVITIES_(UNIT)	_DEVELOPMENT_PERSONNEL(PERSON)	FULL-TIMEPERSONNEL	PART-TIMEPERSONNEL	DOCTOR	MASTER
1	2010	2988	275360	176314	99046	9900	28961
2	2011	3023	327256	218662	108594	11822	34966
3	2012	3742	382057	253493	128564	13342	41509
4	2013	4306	409441	274390	135051	14478	43445
5	2014	5238	432430	285916	146514	16353	49835
6	Agriculture,Forestry,Animal Husbandry and Fishing	34	954	651	303	42	78
7	Mining	69	24116	10471	13645	323	1885
8	Manufacturing	4499	315778	226576	89202	3838	22224
9	Production and Supply of Electric Power and Heat Power	43	2365	990	1375	55	182
10	Construction	56	7617	4874	2743	61	562
11	Traffic,Transport,Storage and Post	13	1408	547	861	11	57
12	Information Transfer, Software and Information Technology Services	64	6209	5321	888	36	592
13	Leasing and Business Services	10	324	200	124	3	15
14	Scientific Research and Technical Service	154	14037	11239	2798	2233	4418
	Management of Water						

Рисунок 5.4 -- Перегляд і редагування наявних даних на сервері

Обробка помилок від будь якого вузла комунікації здійснена через повідомлення у браузері що відображено на рис. 5.5

IsOk є ідентифікатором корректності даних що повертаються від серверу БД або серверу застосунку. Поле Data відображає зміст повідомлення відповідного виключення на стороні серверу застосунку та повністю відповідає сенсу помилки у разі повернення помилки від сервера БД.

Обробка помилок здійснюється стандартними засобами HTML та являє собою вікно елемента Message.Alert.

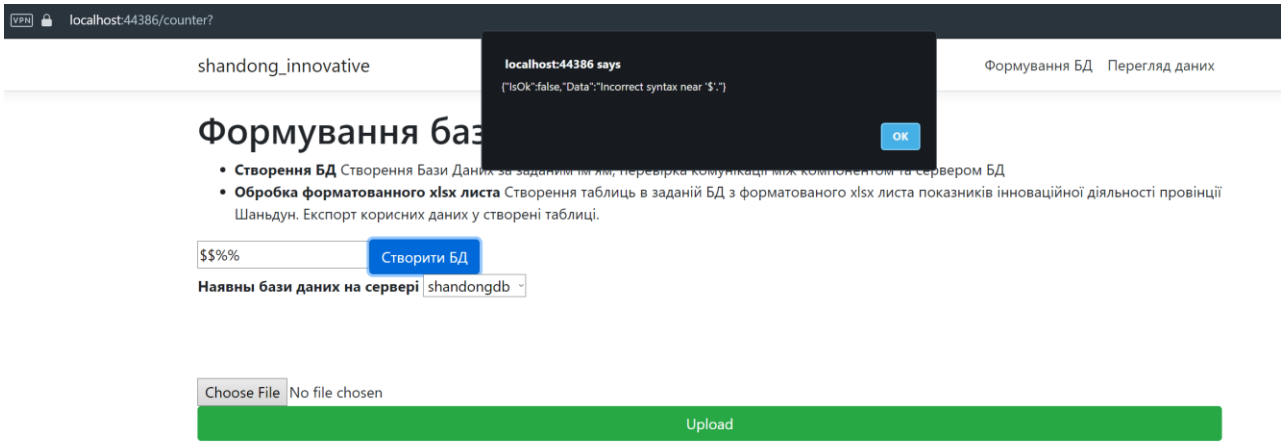


Рисунок 5.5 -- Інтерфейс помилки від сервера бази даних, відображення тексту помилки

На Рис. 5.6 зазначено відображення повідомлення про завантаження інформації про бази даних та таблиці з сервера бази даних.

Реалізовано реактивний алгоритм отримання даних від сервера про наявні бази даних на сервері БД, у разі відключення від сервера додатком, буде відображено відповідне повідомлення та виконано повторне підключення до серверу БД.

Наявні таблиці бази даних на сервері відображаються відповідно до обраної бази. У разі виникнення будь яких помилок на сервері БД буде відображено відповідне повідомлення із кодом помилки та текстом виключення що відправляється на сервер застосунку.

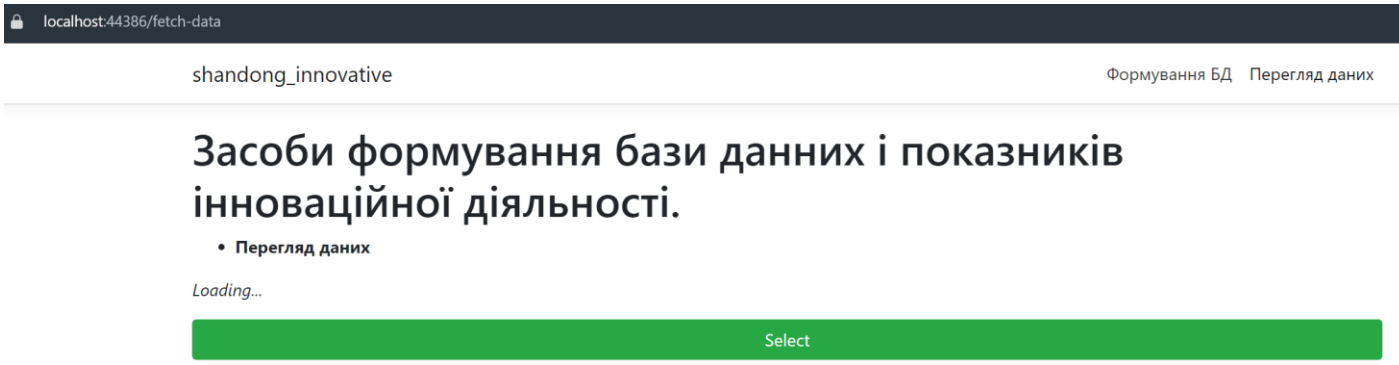


Рисунок 5.6 – Завантаження інформації про бази даних у процесі

5.4 Сценарій роботи користувача з системою

В даному розділі описано основні етапи та моменти реалізації інтерфейсу користувача, який дозволяє взаємодіяти з системою за допомогою веб браузера. Реалізовано можливість переглядати, редагувати та доповнювати дані, що знаходяться в БД, завантажувати дані у вигляді форматованих листів .xls для їх подальшого відображення та редагування.

Інтерфейс реалізовано у вигляді веб-сторінок з використанням HTML/CSS, запити на сервер відправляються за допомогою JSON форматованих даних, комунікація із сервером бази даних здійснюється

лише за наявності активного підключення до сервера, в іншому випадку сторінка відображає повідомлення про завантаження що триває.

Взаємодію клієнтської частини з серверною реалізовано за допомогою використання HTTP-запитів, інформація про взаємодію зберігається та надсилається у форматі JSON документів. Інтерфейс дозволяє оновлювати шаблони документів на сервері.

Таким чином, розроблений інтерфейс дозволяє використовувати можливості серверної частини для генерації документів, перегляду/зміни вмісту бази даних і виконаний в приємній колірній гамі із зручною, інтуїтивно зрозумілою структурою.

ВИСНОВКИ

Була створена система формування бази даних показників інноваційної діяльності провінції Шаньдун, яка створює таблиці показників інноваційної діяльності провінції Шаньдун за посиланням на сервер бази даних, що візуалізовано у вхідних даних листів Excel формату .xls

В результаті розробки програмного продукту було проаналізовано вже існуючі програмні рішення, літературні джерела та обрані методи розробки програмного забезпечення. Було створено і затверджено архітектуру системи що дозволило здійснити програмну реалізацію формування бази даних і таблиць з використанням обраних технологій програмування.

Система дозволяє створювати бази даних на сервері, вводити як експериментальні дані що будуть використовуватися для відображення і редагування так і імпорт корисних даних статистичних показників інноваційної діяльності провінції Шаньдун.

Інформація, зібрана системою і представлена в наочній формі, дозволяє чітко та безпомилково валідувати дані та відображати поточний стан бази даних на сервері БД. Дані можуть бути використані системою для відображення у вигляді таблиць, звітів. Доступ користувачів до системи реалізовано через стандартний веб-браузер.

Основними користувачами системи є керівники наукових організацій, співробітники управлінських органів наукових організацій, співробітники компетентних органів науки на рівні провінції Шаньдун.

ПЕРЕЛІК ПОСИЛАНЬ

1. Організація баз даних: практичний курс: Навч. посіб. для студ. / А. Ю. Берко, О. М. Верес; Нац. ун-т «Львів. політехніка». — Л., 2003. — 149 с. — Бібліогр.: 8 назв.
2. Марко Беллиньясо. Разработка Web-приложений в среде ASP.NET 2.0: задача — проект — решение = ASP.NET 2.0 Website Programming: Problem - Design - Solution. — М.: «Диалектика», 2007. — С. 640. — ISBN 0-7645-8464-2.
3. Database [Електронний ресурс] // Wikipedia. – 2017. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Database.
4. Database [Електронний ресурс] // Oracle FAQ. – 2008. – Режим доступу до ресурсу: http://www.orafaq.com/wiki/Database.
5. PostgreSQL Wiki [Електронний ресурс] // postgresql.org – Режим доступу до ресурсу: https://wiki.postgresql.org/wiki/Main_Page.
6. Introduction to the Oracle Database [Електронний ресурс] // Oracle. – 2017. – Режим доступу до ресурсу: https://docs.oracle.com/cd/B19306_01/server.102/b14220/intro.htm.
7. Шмуллер, Джозеф. Освой самостоятельно UML за 24 часа, 3-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 416 с.
8. C# Language (MSDN) [Електронний ресурс]. — Режим доступу: http://msdn.microsoft.com
9. Фролов А.В. Визуальное проектирование приложений C#. – М.: Кудиц-Образ, 2003. – 512с.
10. Нильсен, Пол. MS SQL Server 2005. Библия пользователя.: Пер. с англ. – М.: ООО «И.Д.Вильямс», 2008. – 1232 с.

11. Хомоненко А.Д. Базы даннях / Хомоненко А.Д., Цыганков В.М. – М.: КОРОНА-Век, 2010. – 736 с
12. Кузин А.В. Базы данных / Кузин А.В. – М.: Академия, 2008. – 320 с.
13. Рихтер Дж. Программирование на платформе Microsoft .NET Framework /Пер. с англ. — 2-е изд., испр. — М.: Издательско-торговый дом «Русская Редакция», 2003 – 512 с.
14. Эндрю Троелсен. Язык программирования C# 2005 (Си Шарп) и платформа .NET 2.0 = Pro C# 2005 and the .NET 2.0 Platform. – 3-е изд. ./ Эндрю Троелсен. – М. : «Вильямс», 2007. – 1168 с.
15. Макконел С. Совершенный код. Мастер-клас / Пер. с англ. – М.: Издательско-торговый дом «Русская редакция»; СПб.: Питер, 2005. – 896 с.Н
16. Элиенс, Антон. Принципы объектно-ориентированной разработки про¬грамм. 2-е издание : Пер. с англ. – М. : Издательский дом «Вильямс», 2002. – 496 с. ил. – Парал. тит. англ.

ДОДАТОК 1

Інструментальні засоби формування бази даних показників інноваційної діяльності провінції Шаньдун

Специфікація

УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТМ4244_20Б

Аркушів 1

2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ІМ. ІГОРЯ СІКОРСЬКОГО»_ТЕ Ф_АПЕПС_ТМ4244_20Б-1	CherednychenkoBI_TM42_ Zapiska.docx	Пояснювальна записка
Комплекс		
Компоненти		
УКР.НТУУ «КПІ ІМ. ІГОРЯ СІКОРСЬКОГО»_ТЕ Ф_АПЕПС_ТМ4244_20Б-2	dod_3.docx	Опис програмного модуля
УКР.НТУУ «КПІ ІМ. ІГОРЯ СІКОРСЬКОГО»_ТЕ Ф_АПЕПС_ТМ4244_20Б-3	Controller.cs	Файл з логікою імпорту даних
УКР.НТУУ «КПІ ІМ. ІГОРЯ СІКОРСЬКОГО»_ТЕ Ф_АПЕПС_ТМ4244_20Б-4	FetchData.js	Front-end файл скрипту
УКР.НТУУ «КПІ ІМ. ІГОРЯ СІКОРСЬКОГО»_ТЕ Ф_АПЕПС_ТМ4244_20Б-5	Counter.js	Controller контроллер

ДОДАТОК 2

Інструментальні засоби формування бази даних показників інноваційної діяльності провінції Шаньдун

Текст програмного модуля

УКР.НТУУ “КПІ”.ТМ4244_20Б-1

Аркушів 23

2020

УКР.НТУУ «КПІ ІМ. ІГОРЯ СІКОРСЬКОГО»_ТЕФ_АПЕПС_ТМ4228_20Б-3

```
 using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;
using OfficeOpenXml;
using shandong_innovative.ViewModels;

namespace borya.Controllers
{
    [ApiController]
    [Route("[controller]")]
    public class WeatherForecastController :
ControllerBase
    {
        private static readonly string[] Summaries = new[]
        {
```

```
            "Freezing", "Bracing", "Chilly", "Cool",
"Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"
        };




        private readonly
ILogger<WeatherForecastController> _logger;


        public
WeatherForecastController(ILogger<WeatherForecastControlle
r> logger)
        {
            _logger = logger;


        }


        [HttpGet]
        public ActionResult Get()
        {


            var result = new List<VmDatabaseData>();


            try
            {
                SqlConnection sqlCon = new
SqlConnection("Data Source=(LocalDb)\\MSSQLLocalDB;Initial
Catalog=shandongdb;Integrated Security=True");


                sqlCon.Open();
```

```
                string query = "SELECT name FROM
master.sys.databases Where database_id > 4";
                SqlCommand cmd = new SqlCommand(query,
sqlCon);


                var rdr = cmd.ExecuteReader();



                while (rdr.Read())
                {
                    result.Add(new VmDatabaseData() {
DatabaseName = rdr.GetString(0) });
                }
                rdr.Close();
                foreach (var r in result)
                {
                    query = $"SELECT name FROM
{r.DatabaseName}.sys.Tables";
                    cmd = new SqlCommand(query, sqlCon);
                    rdr = cmd.ExecuteReader();
                    if (rdr.HasRows)
                    {
                        r.TableNames = new List<string>();
                        while (rdr.Read())
                        {

r.TableNames.Add(rdr.GetString(0));
                        }


                    }
```

```
                    rdr.Close();
                }
                sqlCon.Close();


                return new
JsonResult(result.ToDictionary(x => x.DatabaseName, x =>
x.TableNames));
            }
            catch (Exception ex)
            {
                return new JsonResult(new VmResult() {
IsOk = false, Data = ex.Message });
            }


        }


        private String ColorHexValue(System.Drawing.Color
C)
        {
            return C.A.ToString("X2") + C.R.ToString("X2")
+ C.G.ToString("X2") + C.B.ToString("X2");
        }


        [HttpPost]
        [Route("select")]
        public ActionResult
Select([FromForm]DbSelectViewModel model)
        {
            var result = new List<VmDatabaseData>();


            try
```

```
            {
                SqlConnection sqlCon = new
SqlConnection("Data Source=(LocalDb)\\MSSQLLocalDB;Initial
Catalog=shandongdb;Integrated Security=True");


                sqlCon.Open();
                string query = $"SELECT * FROM
{model.DbName}.dbo.{model.TblName}";
                SqlCommand cmd = new SqlCommand(query,
sqlCon);





                var table = new DataTable();
                using (var rdr = cmd.ExecuteReader())
                {



                        table.BeginLoadData();
                        table.Load(rdr);
                        table.EndLoadData();




                }
                sqlCon.Close();
                var rest =
JsonConvert.SerializeObject(table);
                return new JsonResult(rest);
            }
```

```
            catch (Exception ex)
            {
                return new JsonResult(new VmResult() {
IsOk = false, Data = ex.Message });
            }


        }



        [HttpPost]
        [Route("upload")]
        public ActionResult
Upload([FromForm]FileUploadViewModel model)
        {
            int rowtest;
            int coltest;
            try
            {


                ExcelPackage.LicenseContext =
LicenseContext.NonCommercial;
                using (ExcelPackage package = new
ExcelPackage(model.File.OpenReadStream()))
                {
                    ExcelWorksheet worksheet =
package.Workbook.Worksheets[0];
                    int rowCount =
worksheet.Dimension.Rows;
                    int ColCount =
worksheet.Dimension.Columns;
                    int lastRow = 0;
```

```
                    var headers = new List<string>();
                    var inserts = new Dictionary<int,
List<string>>();
                    bool needContinue = false;
                    for (int row = 1; row <= rowCount;
row++)
                    {
                        for (int col = 1; col <= ColCount;
col++)
                        {
                            rowtest = row;
                            coltest = col;
                            var cell =
worksheet.Cells[row, col].Value;
                            if (cell != null &&
!string.IsNullOrEmpty(cell.ToString().Replace(" ", "")))
                            {
                                if (worksheet.Cells[row,
col].Style.Fill.BackgroundColor.Indexed == 44 && cell !=
null)
                                {
                                    lastRow = row;

headers.Add(cell.ToString().Replace("'", ""));
                                    continue;
                                }
                                if (col >= 2 && row >
lastRow && col <= headers.Count)
                                {
                                    if
(!inserts.ContainsKey(row))
```

```
                                        {
                                            inserts[row] = new
List<string>();

inserts[row].Add(cell.ToString().Replace("'", ""));
                                        }
                                        else
                                        {

inserts[row].Add(cell.ToString().Replace("'", ""));
                                        }


                                    }
                                }
                                else
                                {
                                    if
(inserts.ContainsKey(row) && col <= headers.Count)
                                    {
                                        inserts.Remove(row);
                                        break;
                                    }
                                }
                            }
                        }



                        var splitted =
headers.Skip(1).Select(x => x.Split('\n').Count() == 1 ?
```

```
x.Split('\n')[0] : string.Join("",
x.Split('\n').Skip(2)).Replace(" ", "_"));



                    SqlConnection sqlCon = new
SqlConnection("Data Source=(LocalDb)\\MSSQLLocalDB;Initial
Catalog=shandongdb;Integrated Security=True");


                    sqlCon.Open();

                    string createQuery =
BuildCreateQuery(splitted.ToList(), model);

                    SqlCommand createCommand = new
SqlCommand(createQuery, sqlCon);

                    createCommand.ExecuteNonQuery();


                    SqlCommand insertCommand = new
SqlCommand(BuildInsertQuery(inserts, model), sqlCon);

                    insertCommand.ExecuteNonQuery();



                    sqlCon.Close();


                }


                //  JsonConvert.SerializeObject()

                return new
JsonResult(JsonConvert.SerializeObject(new VmResult() {
IsOk = true, Data = "Success" }));

            }

            catch (Exception ex)

            {
```

```
                return new
JsonResult(JsonConvert.SerializeObject(new VmResult() {
IsOk = false, Data = ex.Message.ToString() }));
            }
        }


        private string BuildInsertQuery(Dictionary<int,
List<string>> data, FileUploadViewModel model)
        {
            foreach (var d in data)
            {
                d.Value[0] = $"'{d.Value[0]}'";
            }
            string query = $"insert into
{model.DbName}.dbo.{CorrectFilename(model.File.FileName)}
values {string.Join(",", data.Select(x =>
$"({string.Join(",", x.Value)} )"))}";
            return query;


        }


        private string CorrectFilename(string Filename)
        {


            return Filename.Replace("-",
"").Replace(".xlsx", "").Replace(" ", "_");
        }
        private string BuildCreateQuery(List<string>
splitted, FileUploadViewModel model)
        {
```

```
            string query = $"Create table
{model.DbName}.dbo.{CorrectFilename(model.File.FileName)}
(Id int NOT NULL PRIMARY KEY Identity(1,1), \"";
            foreach (var t in splitted)
            {
                if (splitted.IndexOf(t) == 0)
                {
                    query += t.Replace("&","" ) + "\"
varchar(255), \"";

                }
                else
                {
                    query += t.Replace("&", "") + "\"
decimal(18,3), \"";
                }

            }

            query = query.Remove(query.LastIndexOf(','),
1);
            query = query.Remove(query.LastIndexOf('"'),
1) + ");";

            return query;
        }

        [HttpPost]
        [Route("test")]
        public ActionResult Test([FromBody] string data)
        {
```

```
            try
            {
                var deserializedData =
JsonConvert.DeserializeObject<VmDb>(data);
                if
(string.IsNullOrEmpty(deserializedData.Name))
                    return new
JsonResult(JsonConvert.SerializeObject(new VmResult() {
IsOk = false, Data = "Name is empty" }));


                SqlConnection sqlCon = new
SqlConnection("Data
Source=(LocalDb)\\MSSQLLocalDB;Integrated Security=True");


                sqlCon.Open();
                string createDbQuery = $"create database
{deserializedData.Name}";
                SqlCommand createCommand = new
SqlCommand(createDbQuery, sqlCon);
                createCommand.ExecuteNonQuery();
                return new
JsonResult(JsonConvert.SerializeObject(new VmResult() {
IsOk = true, Data = "Success" }));
            }
            catch(Exception ex)
            {
                return new
JsonResult(JsonConvert.SerializeObject(new VmResult() {
IsOk = false, Data = ex.Message.ToString() }));
            }
```

```
            }
        }
    }
```

```
import React, { Component } from 'react';
import axios from 'axios';
import ReactDOM from 'react-dom';

//import Select from 'react-select'


export default class Table extends React.Component {

    constructor(props) {
        super(props);
        this.getHeader = this.getHeader.bind(this);
        this.getRowsData = this.getRowsData.bind(this);
        this.getKeys = this.getKeys.bind(this);
    }

    getKeys = function () {
        return Object.keys(this.props.data[0]);
    }

    getHeader = function () {
        var keys = this.getKeys();
        return keys.map((key, index) => {
            return <th key={key}>{key.toUpperCase()}</th>
        })
    }

    getRowsData = function () {
        var items = this.props.data;
        var keys = this.getKeys();
        return items.map((row, index) => {
            return <tr key={index}><RenderRow key={index}
data={row} keys={keys} /></tr>
        })
    }

    render() {
        return (
            <div>
                <table>
                    <thead>
```

```
                        <tr>{this.props.data.length == 0 ?
'' : this.getHeader()}</tr>
                    </thead>
                    <tbody>
                        { this.props.data.length == 0 ? ''
: this.getRowsData()}
                    </tbody>
                </table>
            </div>

        );
    }
}
const RenderRow = (props) => {
    return props.keys.map((key, index) => {
        return <td
key={props.data[key]}>{props.data[key]}</td>
    })
}

class App extends React.Component {
    constructor(props) {
        super(props)
        this.state = {
            selectedView: 'shandongdb'

        }
        const views = this.props.value || {}
    }

    render() {




        const { selectedView } = this.state


        const getMajorMethod = () => {

            const view = this.props.value.filter(({ name
}) => name === selectedView)[0]
            return (
```

```
                <div>
                    {view.minor == null ? <p><em>emty
table list.</em></p> :
                        <select id="table">
                            {view.minor.map(m =>
<option>{m}</option>)}
                        </select>
                    }
                </div>
            )
        }

        let contents = !this.props || !this.props.value ||
this.props == null || this.props.value == null ||
this.props.value.length == 0
            ? <p><em>Loading...</em></p>
            : <div>
                <select id="db" onChange={(e) =>
this.setState({ selectedView: e.target.value })}>
                    {this.props.value.map(({ name }) =>
<option value={name}>{name}</option>)}
                </select>

                {getMajorMethod()}
            </div>
        return (
            <div>{contents}</div>

        )
    }
}




export class FetchData extends Component {
    static displayName = FetchData.name;

    constructor(props) {
        super(props);
        this.state = {
            forecasts: [], loading: false, DbResult: {},
```

```
            modDbResult: [], data: []
        };
        this.getDatabasesData =
this.getDatabasesData.bind(this);
    }
    componentWillMount () {
        this.getDatabasesData();
    }

    getDatabasesData() {
        var self = this;

axios.get('https://localhost:44386/weatherforecast/').
            then(function (response) {
                self.setState({ DbResult: response.data
});
                console.log(response.data);
                var b = [];
                for (const [key, value] of
Object.entries(response.data)) {
                    b.push({ name: key, minor: value });

                }
                self.setState({ modDbResult: b });

            });
        console.log(self.state.DbResult);


        self.state.loading = false;


    }

    static getDatabaseTableNames(argument) {
        console.log(argument);

        return (
            <select
                name="form-field-name"
                options={Object.keys(argument)}
            />
```

```
            //<div>
            //    <p>Select one first</p>
            //    <Select
            //        name="form-field-name"
            //
value={this.state.selectedOption.value}
            //        onChange={this.handleChange1}
            //        options={options1}
            //    />
            //    <p>Then the other</p>
            //    <Select
            //        name="form-field-name"
            //
value={this.state.selectedOption2.value}
            //        onChange={this.handleChange2}
            //        options={filteredOptions}
            //    />
            //</div>

        );
    }


    onClickHandler = () => {

        var apiBaseUrl =
'https://localhost:44386/weatherforecast/select';
        var self = this;
          //  debugger;
            let f = new FormData();

            f.append("DbName",
document.getElementById('db').value);
            f.append("TblName",
document.getElementById('table').value);

            axios.post(apiBaseUrl, f, {
                headers: { 'Content-Type':
'application/json' }
            }).then(function (response) {
                self.setState({ data:
JSON.parse(response.data) });
```

```
            });
        }
        //else {
        //    alert("Please select files first");
        //}

    //}



    render() {





        console.log(this.state.modDbResult);
        let contents2 = !this.state.modDbResult ||
this.state.modDbResult.length == 0
            ? <p><em>Loading...</em></p>
            : <App value={this.state.modDbResult}></App >
       // ReactDOM.render(<Select />,
document.getElementById('root'));

        //document.getElementById('root');



        return (

            <div>
                <h1 id="tabelLabel" >Засоби формування
бази данних і показників інноваційної діяльності.</h1>
                <ul>
                    <li><strong>Перегляд даних</strong>
</li>
                </ul>
```

```
                {contents2}
                <button type="button" className="btn btn-
success btn-block"
onClick={this.onClickHandler}>Select</button>
                <div id="root"></div>

                <Table data={!this.state.data ? [] :
this.state.data} />

            </div>
        );
    }



}
```

УКР.НТУУ «КПІ ІМ. ІГОРЯ СІКОРСЬКОГО»_ТЕФ_АПЕПС_ТМ4244_20Б-5

```
import React, { Component } from 'react';
import axios from 'axios';

export class Counter extends Component {
    static displayName = Counter.name;


    constructor(props) {
        super(props);
        this.state = {
            dbValue: '', file: null, DbResult: {},
loading: true, selectDBValue: '' };


        this.createDb = this.createDb.bind(this);
        this.onChangeHandler =
this.onChangeHandler.bind(this);
        this.handleChange = this.handleChange.bind(this);
        this.getDatabasesData =
this.getDatabasesData.bind(this);
        this.handleDBselected =
this.handleDBselected.bind(this);
    }
```

```
    handleDBselected(e) {
        this.setState({ selectDBValue: e.target.value });
    }

    static getDatabaseTableNames(argument) {
        console.log(argument);

        return (

            <select id="id1">
                {Object.keys(argument).map((k,v) =>
                    <option key={k}>{k}</option>
                )}


            </select>
        );
    }

    componentDidMount() {
        this.getDatabasesData();
    }

    getDatabasesData() {
        var self = this;

axios.get('https://localhost:44386/weatherforecast/').
            then(function (response) {
                self.setState({ DbResult: response.data
});
                console.log(response.data);
            });
        console.log(self.state.DbResult);
        this.state.loading = false;
    }

    createDb() {

        axios({
            method: 'post',
            url:
'https://localhost:44386/weatherforecast/test/',
            data: { Name: this.state.dbValue }
```

```
        }).then(function (resp) { alert(resp.data);
}).catch(err => console.log(err.data));

    }
    onChangeHandler(event) {
        this.setState({
            file: event.target.files[0],
            loaded: 0,
        })

    }
    onClickHandler = () => {

        var apiBaseUrl =
'https://localhost:44386/weatherforecast/upload';
        if (this.state.file != null) {
          //  debugger;
            let f = new FormData();
            f.append("File", this.state.file);
            f.append("DbName",
document.getElementById('id1').value);

            axios.post(apiBaseUrl, f, {
                headers: { 'Content-Type':
'multipart/form-data' }
            }).then(response =>
alert(response.data)).catch(err => console.log(err.data));
        }
        else {
            alert("Please select files first");
        }

    }
    handleChange(event) {
        this.setState({ dbValue: event.target.value });
    }




    render() {
```

```
        let contents = this.state.loading
            ? <p><em>Loading...</em></p>
            :
Counter.getDatabaseTableNames(this.state.DbResult);

        return (
            <div>
                <h1>Формування бази данних</h1>

                <ul>
                    <li><strong>Створення БД</strong>
Створення Бази Даних за заданим ім'ям, перевірка
комунікації між компонентом та сервером БД</li>
                    <li><strong>Обробка форматованного
xlsx листа</strong> Створення таблиць в заданій БД з
форматованого xlsx листа показників інноваційної
діяльності провінції Шаньдун. Експорт корисних даних у
створені таблиці.</li>
                </ul>

                <form>
                    <input type="text"
onChange={this.handleChange}
value={this.state.dbValue}></input>


                    <button className="btn btn-primary"
onClick={this.createDb}>Створити БД</button>
                    <p></p>
                </form>
                <strong> Наявні бази даних на сервері
</strong>
                {contents}
                <div className={this.state.currentCount %
2 === 0 ? 'hidden' : ''}>
                    <p></p><p></p>
                    <br></br><br></br><br></br>
                    <form>
                        <input type="file" name="file"
onChange={this.onChangeHandler} />
                        <button type="button"
className="btn btn-success btn-block"
onClick={this.onClickHandler}>Upload</button>
```

```
                    </form>
                </div>


            </div>
        );
    }
}
```

ДОДАТОК 3

Інструментальні засоби формування бази даних показників інноваційної діяльності провінції Шаньдун

Опис програмного модуля

УКР.НТУУ «КПІ ІМ. ІГОРЯ СІКОРСЬКОГО»_ТЕФ_АПЕПС_ТМ4244_20Б-2

Аркушів 7

2020

АНОТАЦІЯ

В роботі розглянуті теоретичні та практичні аспекти розробки програмного продукту для формування бази даних показників інноваційної діяльності провінції Шаньдун.

Метою роботи є розробка автоматизованої системи максимально адаптованої до вимог провінції Шаньдун та ринкових умов.

Ключові слова: база даних, показники інноваційної діяльності, таблиці, автоматизована система, документообіг, імпорт.

1. ЗАГАЛЬНІ ВІДОМОСТІ

Найменування програмного модуля — “Сервер Імпорту”. Мова програмної реалізації — C#. Середовище розробки — MS Visual Studio 2019.

Даний модуль являє серверний застосунок, який слугує для імпорту даних із форматованих листів показників інноваційної діяльності.

Для використання програмного забезпечення у якості клієнта необхідно :

— комп’ютер, чи смартфон з довільною ОС;

— будь-який браузер або http клієнт.

Для використання програмного забезпечення у якості серверу необхідно :

— комп’ютер з довільною ОС;

— встановлена програмна плафторма .NET framework;

— розгорнута MS SQL база даних.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний модуль надає системі можливість імпортувати дані з xls листів кінцевим користувачам і надавати клієнтській системі механізм для перевірки валідності даних на стороні користувачів.

Основним його призначенням можна вважати створення наступних файлів:

— ViewModel.cs;

— Controller.cs;

— FetchData.js.

Основний функціонал програмного забезпечення, який відповідає за надання веб API є файл FetchData.js:

— generateKey(req, res) — відповідає за генерацію секретного ключа для користувача.

— validateXls (req, res) — відповідає за початкове підтвердження користувачем отримання секретного ключа і завершення його реєстрації.

— submitXls(req, res) — відповідає за валідацію даних.

3. ВХІДНІ ДАНІ

Вхідними даними для роботи програмної системи є форматований xls файл.

Вихідними даними є:

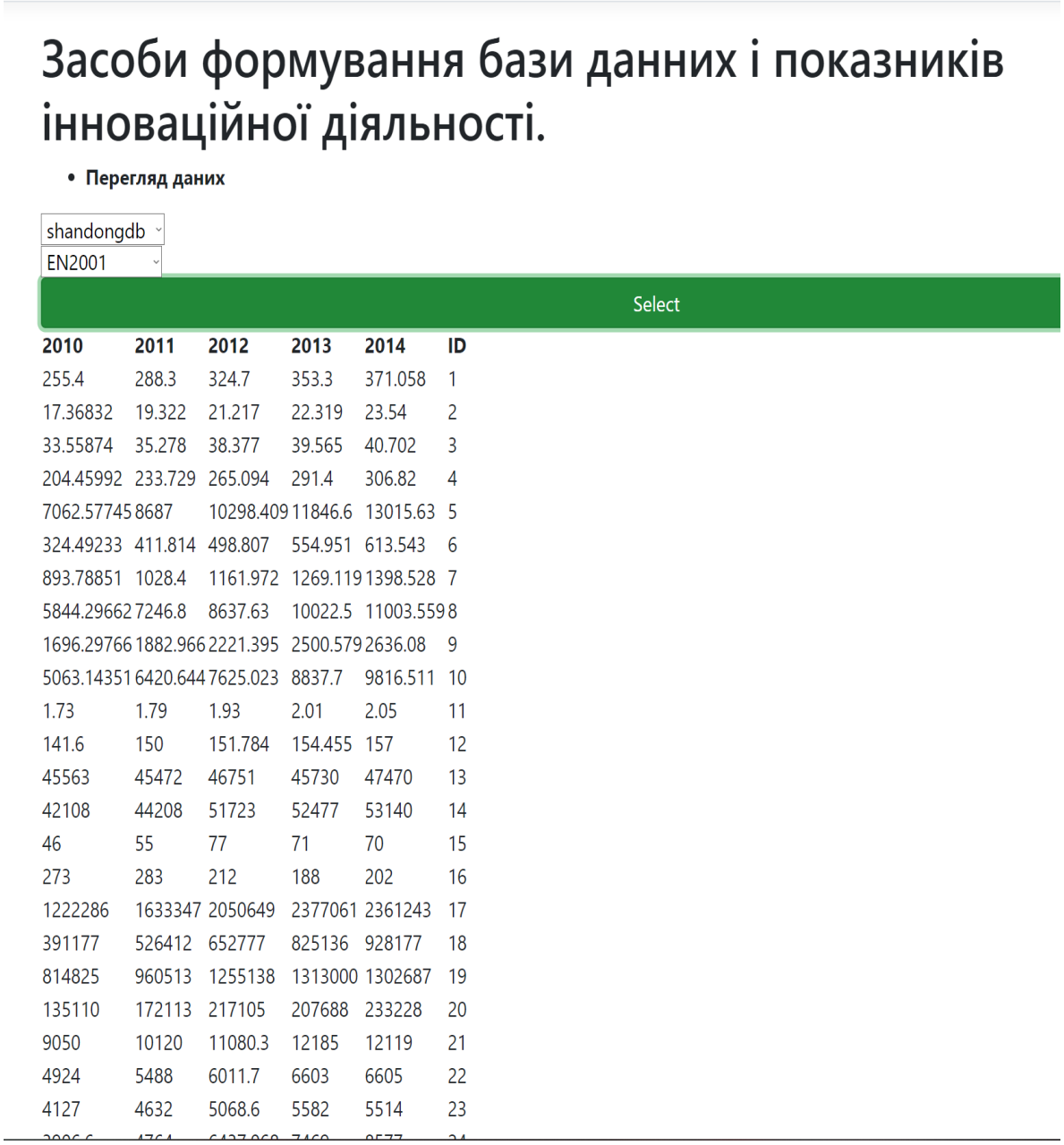
— Корисні дані на сервері БД.

— Результат імпорту даних на сервер БД.

Рисунок 3.1 — Запит на отримання даних з сервера

4. ВИХІДНІ ДАНІ

Вихідними даними результату роботи програмної системи є веб-сторінка, на якій для користувача зображено дані імпортовані на сервер БД.



Засоби формування бази данних і показників інноваційної діяльності.

- Перегляд даних

shandongdb

EN2001

Select

2010	2011	2012	2013	2014	ID
255.4	288.3	324.7	353.3	371.058	1
17.36832	19.322	21.217	22.319	23.54	2
33.55874	35.278	38.377	39.565	40.702	3
204.45992	233.729	265.094	291.4	306.82	4
7062.57745	8687	10298.409	11846.6	13015.63	5
324.49233	411.814	498.807	554.951	613.543	6
893.78851	1028.4	1161.972	1269.119	1398.528	7
5844.29662	7246.8	8637.63	10022.5	11003.559	8
1696.29766	1882.966	2221.395	2500.579	2636.08	9
5063.14351	6420.644	7625.023	8837.7	9816.511	10
1.73	1.79	1.93	2.01	2.05	11
141.6	150	151.784	154.455	157	12
45563	45472	46751	45730	47470	13
42108	44208	51723	52477	53140	14
46	55	77	71	70	15
273	283	212	188	202	16
1222286	1633347	2050649	2377061	2361243	17
391177	526412	652777	825136	928177	18
814825	960513	1255138	1313000	1302687	19
135110	172113	217105	207688	233228	20
9050	10120	11080.3	12185	12119	21
4924	5488	6011.7	6603	6605	22
4127	4632	5068.6	5582	5514	23

Рисунок 4.1 — Вікно з даними на сервері БД